НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Аналітична система підтримки терапевта»

Виконав (-ла):
студент (-ка) IV курсу, групи ІА-61
Бойко Дмитро Васильович _____

Керівник:
к.т.н., доц. Дорогий Я.Ю. _____

Рецензент:
Проф. каф. ОТ НТУУ
«КПІ ім. Ігоря Сікорського», д.т.н.
Цуркан Василь Васильович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»
Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту
Бойку Дмитру Васильовичу

1. Тема проєкту «Аналітична система підтримки терапевта», керівник проєкту к.т.н., доц. Дорогий Я.Ю. , затверджені наказом по університету від «07» травня 2020р. №1081-с
2. Термін подання студентом проєкту _____
3. Вихідні дані до роботи: науково-технічна література з проектування систем розробки програмних механізмів, документація існуючих систем, публікації у періодичних виданнях і матеріали з мережі Інтернет за темою роботи.
4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити): програмний модуль для модифікації зображень; програмний модуль для графічної обробки та класифікації об’єктів на зображені; програмний модуля для створення математичної моделі зображення; локалізований та зрозумілий інтерфейс користувача; функціонал збереження результатів дослідження.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): діаграма прецедентів використання, діаграма класів, діаграма послідовностей, діаграма компонентів.

6. Дата видачі завдання 12.03.2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення ззавданням	12.03.2020	
2	Аналіз існуючих аналогів	26.03.2020	
3	Дослідження альтернативних рішень	26.03.2020	
4	Проектування компонент системи	05.05.2020	
5	Розробка протоколу	15.05.2020	
6	Розробка схем, ілюстрацій	22.05.2020	
7	Оформлення документації	02.06.2020	
8	Захист дипломного проекту	17.06.2020	

Студент Дмитро Бойко

Керівник Ярослав Дорогий

АНОТАЦІЯ

Бойко Д.В. Аналітична система підтримки терапевта КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 73 с. тексту, 29 рисунків, 4 таблиці, посилання на 22 літературне джерело, 4 додатки.

Об’єктом розробки є аналітична система підтримки терапевта

Мета розробки – розробка програмного комплексу для користувачів- лікарів, що дозволить підвищити точність аналізу медичних зображень та дозволить експортувати результати медичних дослідів.

Даний дипломний проект присвячено розробці клієнтського додатку для модифікації та обробки медичних зображень. Метою бакалаврської роботи є створення системи з унікальним функціоналом, за допомогою якого лікарі зможуть застосувати методи математичного аналізу та обробку зображень у діагностиці.

В дипломному проекті розглянуті основні технології розробки програмного забезпечення для створення сучасного, функціонального додатку, їх переваги та недоліків.

ANNOTATION

Boyko D.V. Analytical support system for therapist. Igor Sikorsky KPI, Kyiv, 2020.

The project contains 72 pages. text, 31 figures, 4 tables, references to 22 literature sources, 4 appendices.

The object of development is an analytical support system for the therapist .

The purpose of the development is to develop a software package for medical users, which will increase the accuracy of medical image analysis and will export the results of medical experiments. This diploma project is devoted to the development of a client application for the modification and processing of medical images. The purpose of the bachelor's thesis is to create a system with unique functionality, through which doctors can apply the methods of mathematical analysis and image processing in diagnostics.

The diploma project considers the main technologies of software development for creating a modern, functional application, their advantages and disadvantages.

Номер сторінк	Формат	Позначення	Найменування	Кільк. листів	Номер екзем.	Приміт-ка
1			Документація загальна			
2						
3			Заново розроблена			
4	А4	ІА61.030БАК.002 ПЗ	Пояснювальна записка	73		
5						
6	А3	ІА61.030БАК.003 Д1	Діаграма прецедентів	1		
7						
8	А3	ІА61.030БАК .004 Д2	Діаграма класів	1		
9						
10	А3	ІА61.030БАК.005 Д3	Діаграма послідовностей	1		
11						
12	А3	ІА61.030БАК.006 Д4	Діаграма компонентів	1		
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Арк.	№ докум.	Підпис	Дата	ІА61.030БАК.005 ТП		
Розробив		Бойко Д.В.			Аналітична система підтримки терапевта Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Дорогий Я.Ю.				Т	
Рецен.						*«КПІ ім. Ігоря Сікорського», ФІОТ, ІА-61*	
Н. Контр.							
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Аналітична система підтримки терапевта»

Київ – 2020 року

ЗМІСТ

ВСТУП

Актуальність проблеми. Останнім часом одним з актуальних напрямків розвитку комп'ютерних технологій в медицині стає обробка цифрових зображень: поліпшення якості зображення, відновлення пошкоджених зображень, розпізнавання окремих елементів. Розпізнавання патологічних процесів є однією з найбільш важливих завдань обробки та аналізу медичних зображень. При вирішенні задач розпізнавання все частіше використовуються комп'ютерні системи діагностики - computer aided diagnostic (CAD).

Об’єктом дослідження є алгоритми та засоби модифікації та аналізу зображень.

Предметом дослідження є методи графічної модифікації та створення математичних моделей об’єктів інтересу на зображенні.

Метою даного проекту є розробка програмного комплексу для користувачів-лікарів, що дозволить підвищити точність аналізу медичних зображень та дозволить експортувати результати медичних дослідів. Це також сприяє спрощенню взаємодії лікарів у питаннях аналізу та порівнянню різних діагностичних випадків у практиці.

Завданням є створення клієнтської програми – додатку для настільних комп’ютерів – системи допомоги прийняття рішень у фізіотерапії. Для цього в рамках дипломного проекту повинні бути розроблені такі підсистеми:

– програмний модуль для модифікації зображень;

– програмний модуль для графічної обробки та класифікації об’єктів на зображені;

– програмний модуля для створення математичної моделі зображення;

– локалізований та зрозумілий інтерфейс користувача;

– програмний модуль збереження результатів дослідження.

Зберігання даних в файли з спеціальним форматом гарантує забезпечення аналізу даних в різних системах та програмах, можливість розподіленого і одночасного доступу до них. Зазвичай, файли математичних моделей подібних програм зберігають у файлах, які важко пошкодити та змінити. Для медичних цілей я бачу доцільним використати файли формату roi, а якщо конфігурація передбачає декілька математичних об'єктів, то автоматично архівізувати їху форматі rar.

Методи дослідження. Було проаналізовано значний теоретичний матеріал у друкованих джерелах та мережі Інтернет. Порівняно значну кількість наявних рішень, та синтезовано найбільш доцільний перелік функціоналу, необхідний для додатків, які вирішують поставлені перед ними користувачами-лікарямизадачі.

Практичне значення. Розроблювані модулі можна використовувати як альтернативу існуючім рішенням, а також для створення більш розвинутих систем на основі існуючого функціоналу.

На захист виносяться такі положення:

– теоретичне обґрунтування необхідності розробки програмного продукту;

– визначення понять модифікації і обробки зображень;

– цілі, завдання, функції та основні принципи побудови додатку.

Структура роботи. У розділі 1 виконаний огляд предметної області та аналіз існуючих рішень.

У розділі 2 виконаний огляд та порівняння існуючих програмних технологій, що можуть бути використані для реалізації проекту, проаналізовано та обрано найбільш доцільні з них відповідно до завдання.

У розділі 3 розглянуті інструменти та засоби розробки, які було використано при розробці програмного продукту.

У розділі 4 визначено необхідні дії для запуску програми та розглянуто основний її функціонал.

1 АНАЛІЗ КОМП'ЮТЕРНОЇ ОБРОБКИ ЗОБРАЖЕНЬ

1.1 Огляд напрямів обробки зображень

Обробка зображення - це спосіб перетворення зображення в цифрову форму та виконання деяких операцій, щоб отримати розширене зображення або витягнути з неї якусь корисну інформацію. Це тип розподілу сигналу, в якому вхідний образ, як відео кадр або фотографія, і вихід може являти собою зображення або характеристики, пов'язані з цим зображенням. Зазвичай система обробки зображень включає обробку зображень як двовимірних сигналів при застосуванні вже встановлених методів обробки сигналів [1].

Ця галузь є однією із швидкозростаючих сьогодні, з її застосуванням у різних аспектах бізнесу. Обробка зображень утворює основні напрями досліджень в інженерних та комп'ютерних дисциплінах.

Обробка обробки зображень в основному включає в себе наступні три етапи:

– імпортування зображення за допомогою оптичного сканера або цифрової фотографії.

– аналіз і маніпулювання зображенням, що включає в себе стиснення даних та поліпшення зображень та виявлення шаблонів, які не є на очах людей, як супутникові фотографії.

– вихід - це останній етап, в результаті якого може бути змінений образ або звіт, який базується на аналізі зображень [2].

Існує два типи обробки зображень - аналоговий та цифровий. Аналогові або візуальні методи обробки зображень можуть бути використані для друкованих копій, як роздруківки та фотографії. Аналітики зображень використовують різні засади інтерпретації, використовуючи ці візуальні методи. Обробка зображення не обмежується просто областю, яку слід вивчати, але й знанням аналітика. Асоціація є ще одним важливим інструментом обробки зображень за допомогою візуальних технологій. Тому

аналітики застосовують комбінацію особистих знань та допоміжних даних для обробки зображень [3]. Технології цифрової обробки допомагають маніпулювати цифровими зображеннями за допомогою комп'ютерів. Оскільки вихідні дані з датчиків зображення від супутникової платформи містять недоліки. Щоб подолати такі недоліки та отримати оригінальність інформації, вона повинна проходити різні етапи обробки.

Три загальні етапи, до яких під час використання цифрової техніки повинні піддаватися всі типи даних, є попередня обробка, розширення та відображення, видобування інформації [4].

Напрямки обробки зображень:

- візуалізація;
- покращення та відновлення графіки;
- редагування;
- математичне моделювання та перетворення;
- розпізнавання об'єктів на зображенні.

1.1.1 Візуалізація

Візуалізація – це процес подання даних графічно та взаємодію з цими поданнями, щоб отримати розуміння даних. Традиційно, комп'ютерна графіка забезпечила потужний механізм для створення, маніпулювання та взаємодії з цими представленнями.

Відображення результатів – такі методи, як відображення контурів, тензорних потокових потоків та багато інших, використовуються для створення образів, що відображають поведінку скалярного, векторного, тензорного або багатовимірного стану поведінки.

Модель трансформації - чим складніші моделі, тим важливішим стає можливість інтерактивно змінювати вид моделі, а також інші атрибути

відображення, такі як перспектива, відстань та положення спостерігача. Залежно від програмного та апаратного середовища ці можливості можуть варіюватися від зміни статичних параметрів зображення до 3D-перетворення даних результату в реальному часі.

Анімація - значна частина фактичної поведінки включає в себе рух або варіації в часі. В результаті візуалізації, анімація включає в себе дві основні проблеми: відображення поведінки з часом; і навігацію або "прогулянку" моделі в 3-D просторі. Анімація - це відносно молодий і апаратно-залежний інструмент візуалізації, який потребує методів швидкого перемальовування або зберігання та відтворення окремих "кадрів" візуальної інформації.

Результат польової маніпуляції - окрім анімації фіксованої, відома кількість результатів, новіший інтерфейс користувача та технології відображення дозволяють інтерактивно вивчати природу поля результату. Такі методи включають зондування 3-D місць в межах моделі для відображення даних поля в даній точці, інтерактивне відображення ізолів та рівнів результатів, обрізання обсягу та нарізання, а також швидке порівняння декількох станів поведінки [5].

1.1.2 Покращення та відновлення графіки

Метою відновлення зображення є "компенсувати" або "скасувати" дефекти, які погіршують зображення. Деградація набуває багатьох форм, таких як розмиття руху, шум та неправильне фокусування камери. У таких випадках, як розмиття руху, можна дуже добре оцінити фактичну функцію розмиття та "відмінити" розмиття, щоб відновити оригінальне зображення. У випадках, коли зображення пошкоджене шумом, краще, що ми можемо сподіватися, це компенсувати деградацію, яку вона спричиняє.

Методи компенсації шуму в порядку зростання ефективності:

– зворотня фільтрація – для випадків, коли ми знаємо функцію розмиття;

– фільтрація Вайнера – для випадків, коли ми маємо математичне очікування або приблизно знаємо функцію розмиття; згладжує шуми;

– вейвлет реставрація;

– сліпа деконволюція – для випадків, коли ми нічого не знаємо про характер шумів та розмиття [6].

1.1.3 Математичне моделювання та перетворення

Метою моделювання чи представлення зображення є пошук правильних способів математичного опису та аналізу зображень. Саме тому це найважливіший крок у обробці зображень. Одне, однак, варто усвідомити, що не існує абсолютно найкращих представлень, оскільки

оптимальність неминуче залежить від конкретних завдань обробки, як у випадку різних поданнях натуральних чисел: десяткова система є більш зручною, ніж дядічна в повсякденному житті, але остання є більш природньою для цифрових чи квантових комп'ютерів.

Існують такі моделі представлення зображень:

– одномірні моделі часових серій;

– випадково-польові моделі;

– синтаксичні моделі – процедурна модель, як "будь-який процес, який генерує або розпізнає образи, клас, який він визначає, складається з зображень, які він приймає". Зазвичай мова визначається набором рядків над алфавітом, де алфавіт складається з безлічі всіх символів, які можуть з'являтися в рядках мови, а рядок - це кінцева упорядкована послідовність символів. Граматика являє собою набір правил, які визначають, як формуються рядки мови. Таким же чином, граматика може використовуватися для розпізнавання рядків мови за допомогою правил у зворотному порядку. Масив граматики генерує зображення, багатора-

зово замінюючи підмножини іншими підмножинами. Граматика стохастичних масивів – це така, в якій правила заміни є імовірнісними;

– регіональні моделі - регіональні моделі визначаються за допомогою регіонів, а не пікселів, як примітивів. Дана модель визначає форми регіонів і дає правила для їх розміщення в площині, тим самим дозволяючи підвищити контроль над деякими характеристиками моделі. І форми, і правила розміщення можуть бути задані статистично;

– моделі високого рівня – розглядають зображення як більше, ніж просто просторову варіацію. Вони дозволяють інтерпретувати зображення в термінах цілей аналізу та контексту області даних [7].

1.1.4 Редагування

Редагування зображень означає зміну або покращення цифрових або традиційних фотографічних зображень за допомогою різних методів та інструментів. Редагування зображень здійснюється для створення найкращого вигляду зображень, а також для поліпшення загальної якості зображення за різними параметрами. Редагування зображень здійснюється для видалення небажаних елементів, регулювання геометрії зображення, обертання та обрізання, зміни кольорів або додавання спеціальних ефектів до зображення.

1.1.5 Розпізнавання та класифікація об’єктів

Розпізнавання образів стосується передусім опису та класифікації вимірювань. Багато різних математичних методів, що використовуються для вирішення проблем розпізнавання образів, можна згрупувати за двома загальними підходами. Це вирішально-теоретичний (або дискримінаційний) підхід та синтаксичний (або структурний) підхід.

У вирішально-теоретичному рішенні набір характеристичних

вимірювань, що називається функціями, витягується з шаблонів. Кожен шаблон представлений вектором властивостей, і розпізнавання кожного шаблону, як правило, здійснюється шляхом розподілу об'єкта.

З іншого боку, в синтаксичному підході кожен зразок виражається як склад його компонентів, що називається subpatterns або шаблони примітивів. Цей підхід аналізує структуру шаблонів і синтаксис мови. Розпізнавання кожного шаблону зазвичай здійснюється шляхом аналізу структури шаблону відповідно до заданого набору правил синтаксису. У деяких програмах обидва ці підходи можуть бути використані. Наприклад, в задачі, що стосується складних патернів, теоретико-методологічний підхід, як правило, ефективний при визначенні примітивних шаблонів, а потім синтаксичний підхід використовується для розпізнавання примітивів та самого шаблону.

Мета вилучення особливостей полягає в тому, щоб зменшити зображення змінного розміру до фіксованого набору візуальних функцій. Модель класифікації зображень, як правило, будується за допомогою методів вилучення характеристик зображень. Незалежно від того, базуються вони на традиційних підходах для комп'ютерного бачення, таких як, наприклад, наближені фільтри, методи гістограми тощо, або глибокі методи навчання, всі вони мають однакову мету: витягувати функції з вхідного зображення, які є репрезентативними для завдання в руках і використовуйте ці функції для визначення класу зображення.

1.2. Аналіз існуючих рішень

1.2.1 JMicroVision

JMicroVision - інструмент для аналізу зображень і для вимірювання і кількісного вираження характеристик елементів цих зображень. Програма містить більшість поширених операцій обробки зображень, має ефективну систему візуалізації та інноваційні функції. Вона містить інструменти для кількісної оцінки зображень в ручному і автоматичному режимі.

JMicroVision має простий і зрозумілий призначений для користувача інтерфейс з потужними функціями. Для її використання не потрібно бути фахівцем в області аналізу зображень.

JMicroVision була розроблена спеціально для аналізу зображень шліфів гірських порід, але вона може бути легко використана і в інших областях прикладних досліджень, де потрібно подібна обробка зображень.

Надалі ці цифрові зображення можуть бути піддані спеціальній обробці для підвищення якості зображення, його інформативності, а також для проведення різних вимірів. Ця обробка може проводитися як за допомогою універсальних графічних пакетів типу фоторедакторів, так і за допомогою спеціалізованих програм.

Основні властивості програми:

– завантаження зображень в TIFF, BMP, FlashPiX, GIF, JPEG, PNG, і PNM форматі;

– ефективна система візуалізації;

– кількісний аналіз компонентів зображення: виділених об'єктів або фону

– морфологічний аналіз об'єктів (розмір, форма, орієнтація, текстура і т.п.);

– класифікація виділених об'єктів;

– візуальна обробка зображень (бінарні та морфологічні операції, фільтрація, сегментація і т.п.);

– корекція і очищення зображень (геометрична коригування за допомогою контрольних точок);

– чисельний підрахунок виділених точок на зображенні;

– інструменти для збору даних в одному або двох вимірах;

– створення анотацій для зображень і карток описів; побудова вимірювальних профілів, які фіксують зміни гранулометричних параметрів і графічної щільності виділених об'єктів або фону;

– збереження всіх вимірювань, вихідних даних, калібрувань і налаштувань в одному файлі проекту.

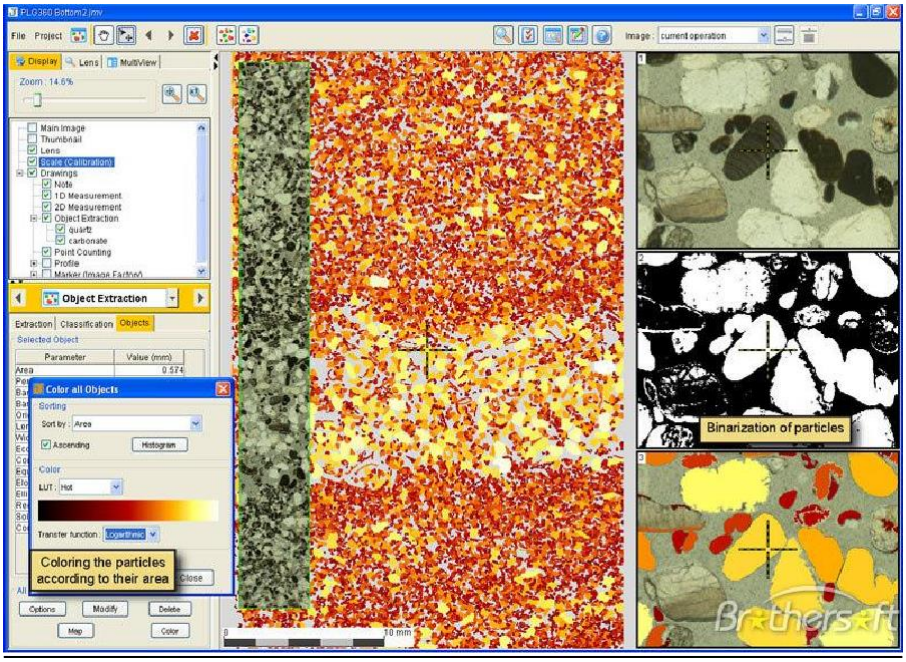


Рисунок 1.1 – Вікно програми JMicroVision під час роботи

Дана програма має наступні характеристики та вимоги до системи.

Таблиця 1.1 – Характеристики програми JMicroVision

Програмні платформи	Windows, Solaris та Linux
Операційні системи	ОС Windows, ОС Linux, Solaris та Mac OS
Необхідна ОЗУ	256 Мб ОЗУ (512 Мб для обробки великих зображень)
Необхідний дисковий простір	60-90 Мб

1.2.2 Altami Studio

Програма Altami Studio розроблена для захоплення, дослідження і обробки зображень, а також для проведення вимірювань. Завдання програми – допомогти користувачеві проаналізувати зображення, отримане з обладнання, зробити висновки, підготувати і зберегти результати дослідження.

Зображення можуть бути отримані за допомогою різного дослідного устаткування: мікроскопів, телескопів, рентгенівських і інших апаратів. Спостереження і вимірювання відбуваються в режимі реального часу (зображення отримують з камери або інших захоплюючих пристроїв).

Особливості програми:

– вимірювання та обробка зображень в реальному часі;

– інтеграція пристроїв захоплення;

– широкі можливості вимірювання та розмітки;

– численні растрові операції;

– автоматичний пошук об'єктів;

– всі дані в одному документі;

– швидке створення звіту або створення звіту за допомогою редактора;

– підтримка сесій.

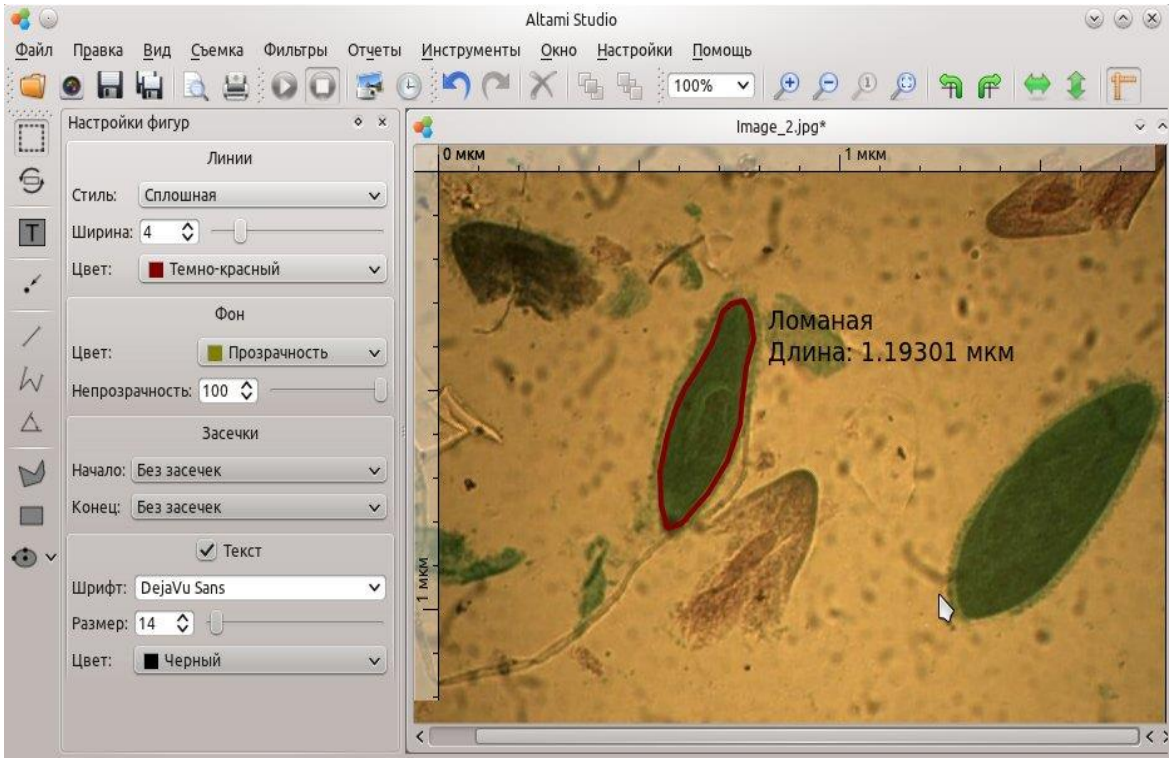


Рисунок 1.2 – Вікно програми Atlami Studio під час роботи.

Характерна особливість Altami Studio від більшості класичних програм - повна підтримка пристроїв захоплення. Altami Studio дозволяє не тільки захоплювати кадри з камери, але і управляти її настройками. Управління настройками камери дає можливість зберігати їх разом з документами і експортувати до звіту для того, щоб не повторювати одні й ті ж дії з налаштування камери кілька разів. Крім того, всі можливі дії зі статичною картинкою застосовні і до потоку з камери. Це безсумнівна перевага програми. Не варто також забувати про широкий спектр підтримуваних програмою в різних операційних системах стандартів пристроїв захоплення, включаючи Microsoft DirectShow, UVC, unicap, Qt Capture. Додатком Altami Studio підтримуються моделі камер Canon EOS і Canon PowerShot, а також високошвидкісні USB 3.0 відеокамери, які працюють за даним протоколом, Point-Grey. Дана програма має такі характеристики та вимоги до системи.

Таблиця 1.2 – Характеристики програми Atlami Studio

Операційні системи	Windows XP SP3, Windows Vista, Windows 7 (архітектури x86 и x64); операційні системи на базі ядра Linux: Alt Linux, Open Suse 11.3, Ubuntu 10.04 LTS та вище; операційна система Mac OS X 10.6
ОЗУ	оперативна пам'ять 1 GB, рекомендований обсяг оперативної пам'яті - 2 GB і вище
Необхідний дисковий простір	1 GB
Процесор	процесор Intel з тактовою частотою від 2 ГГц, рекомендується двоядерний процесор Intel з тактовою частотою від 1,6 ГГц; можливе використання процесора інших виробників з аналогічною продуктивністю
Графіка	дисплей 1024x768, рекомендований - 1280x1024 і вище; графічний адаптер повинен забезпечувати роботу в TrueColor режимі (24 або 32 біта на піксель)
USB-порти	для роботи з камерами необхідна наявність вільного USB-порту на окремому USB-хаб для кожної камери

1.2.3 NEXSYS ImageExpert™ Pro 3

Програма NEXSYS ImageExpert Pro 3 призначена для вирішення завдань кількісного аналізу зображень мікроструктур в металографії, матеріалів і порошків в матеріалознавстві, препаратів і об'єктів в медицині та біології.

Аналізатор дозволяє отримувати широкий спектр геометричних параметрів елементів структури, до найбільш важливим з яких можна віднести процентні частки складових; площі; периметри; мінімальні, максимальні і середні діаметри; параметри форми і витягнутості об'єктів; характеристики розподілу об'єктів (в тому числі локальні діаграми і діаграми вільних відстаней, гістограми міжцентровою відстаней і відстаней між об'єктами); характеристики анізотропії структур і багато іншого. Отримувані характеристики доступні як для кожного об'єкта окремо, так і у вигляді їх статистичної вибірки. Аналізатор дозволяє представляти отримані розподілу параметрів відповідно до вимог російських і міжнародних стандартів. Будучи універсальним інструментом, NEXSYS ImageExpert Pro 3 використовує настройки стандартів не тільки включені в поставку, а й дозволяє користувачам самостійно налаштовувати аналізатор на роботу відповідно до вимог потрібної нормативної документації.

Особливості програми:

– засоби для роботи з відеокамерою;

– реалізація методу пошарової мікроскопії (розширеного фокуса);

– тривимірна візуалізація з використанням технології openl;

– калібрування оптичної системи комплексу;

– накладення масштабної сітки на зображення; циклічний режим скасування перетворень відкат-повернення;

– динамічний режим попереднього перегляду для більшості методів;

– розширена номенклатураметодів;

– бінаризація зображень;

– сегментація зображень за кольором;

– фільтрація об'єктів по геометричним параметрам;

– комплексне відображення результатів;

– налаштування відображаються результатів;

– формування автозвіту.

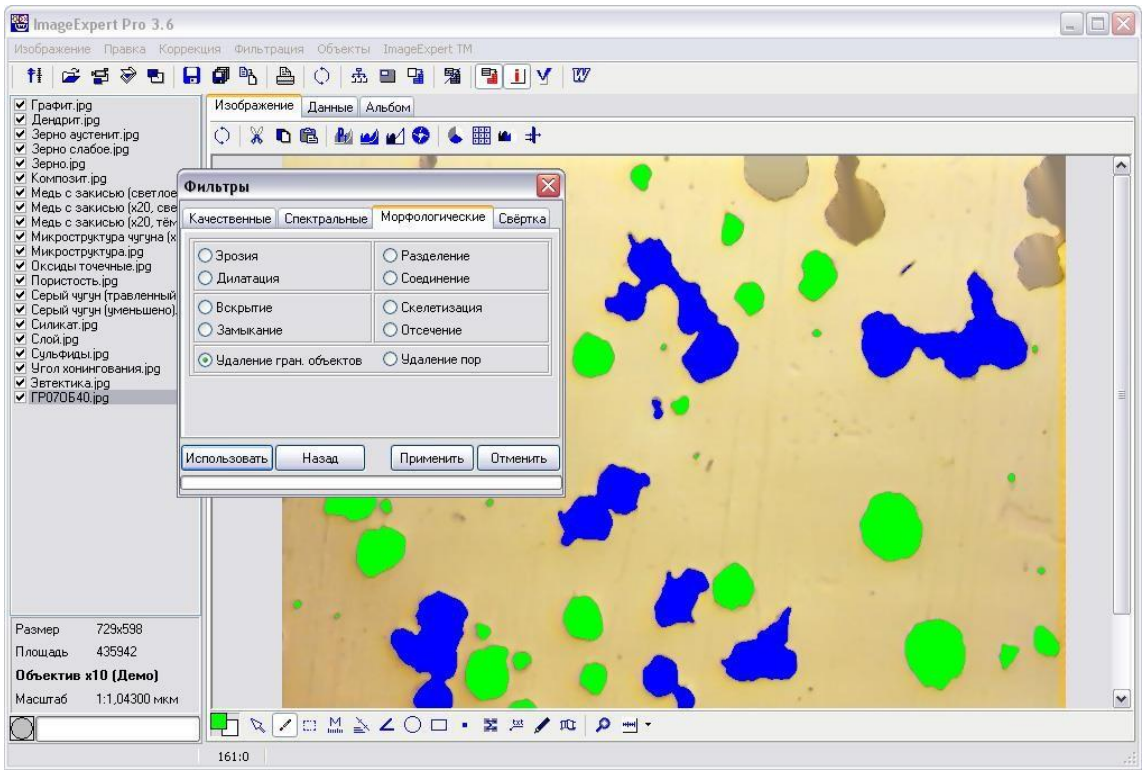


Рисунок 1.3 – Вікно програми NEXSYS ImageExpert Pro 3 під час роботи

Для роботи апаратно-програмного комплексу NEXSYS ImageExpert можна використовувати будь-який сучасний настільний комп'ютер, пропонований

комп'ютерними магазинами – ніяких особливих вимог до сучасних комп'ютерів немає. Відеоадаптер може бути як вбудованим в материнську плату або в процесор, так і окремим пристроєм. Вибір всіх параметрів нового комп'ютера визначається власних уподобань. Дана програма має наступні характеристики та вимоги до системи.

Таблиця 1.3 – Характеристики програми NEXSYS ImageExpert Pro 3

Операційні системи	Microsoft Windows 7 32/64-bit
ОЗУ	оперативна пам'ять DDR3 від 2 GB до 8 GB
Необхідний дисковий простір	320-750 GB
Процесор	Процесори класу Intel (Core2Duo, i3-i7) / AMD (Athlon / Phenom II, Athlon A4-A8)
Графіка	дисплей 1440x900 і вище
USB-порти	Три вільних порти USB 2.0 / 3.0
Оптичні приводи	DVD+RW

Висновки до розділу 1

В даному розділі були оглянуті підходи, напрями та основні види модифікації та обробки зображень. Також були оглянуті існуючи рішення і додатки, що можна знайти у мережі Інтернет. Було опрацьовано теоретичні відомості для реалізації додатку та всієї необхідної для розробки інформації.

2 ВИБІР ТА ОБГРУНТУВАННЯ ПРИНЦИПУ ПОБУДОВИ СИСТЕМИ

2.1 Принципи побудови медико-аналітичних систем

В наш час особливе значення має застосування систем підтримки прийняття рішень (СППР) в ситуаціях які вимагають аналітичного розгляду. СППР є комп'ютерною системою, яка надає допомогу в прийнятті рішень. Процес прийняття рішення безпосередньо пов'язаний з обробкою і структуруванням великих обсягів інформації. СППР - програмний комплекс з інструментальних засобів, які можуть обробляти вхідні дані, здійснювати моделювання, прогнозування та приймати рішення. Комплексний аналіз розглядає масу критеріїв, оцінює ситуацію і результат кожного рішення, пропонуючи і консультуючи по кожному обраному результату. СППР виникло завдяки з'єднанню управлінських інформаційних систем і систем управління базами даних. Завдяки СППР стала можлива оптимізація і ранжування рішень. Можна як вибрати з безлічі випадків прийнятого рішення найбільш підходящий результат так і зовсім виключити зайві результати.

Велика кількість чинників і ознак які викликають хворобу, ускладнюють прийняття вірного рішення в медицині. Важливо правильно визначити фактор викликав недуга. Так як часто це буває комплекс факторів.

Через великий потік інформації лікар – діагност часто не має можливості якісно повністю її обробити. З даною проблемою якраз відмінно справляється обчислювальна техніка, яка має необхідне програмне забезпечення.

Так як прийняті рішення прийняті в медицині безпосередньо впливають на життя і здоров'я людини, а для складних задач діагностики СППР поки що не є доцільними, оптимальною стратегією є надання лікарю-діагносту потрібного для математичного аналізу і модифікації медичних зображень інструментарія, що поєднає точність комп'ютерного аналізу і обчислень з досвідом лікаря і його здатністю до прийняття складних рішень з урахуванням контексту.

При впровадженні інформаційних систем і СППР актуальною є проблема їх функціональності і придатності, яка виражається в тому, що чим більш функціональна система, тим вона більш складна, а значить, і менш придатна для практичного застосування.

Також, не менш важливим є надання можливості розширювати та доповнювати систему, а також не концентрувати систему на конкретній задачі, що звузить поле її застосування до конкретної предметної області.

З точки зору подальшої розробки, необхідно реалізувати механізм розширення системи за допомогою нових плагінів.

Варте уваги питання універсальності програмного продукту та продуктивності його роботи у різних системах. Зазвичай, подібні системи можуть працювати на різних операційних системах, але на практиці частіше за все обирають як основну одну із популярних операційних систем.

На рисунку, наведеному нижче, представлений приклад організації СППР.

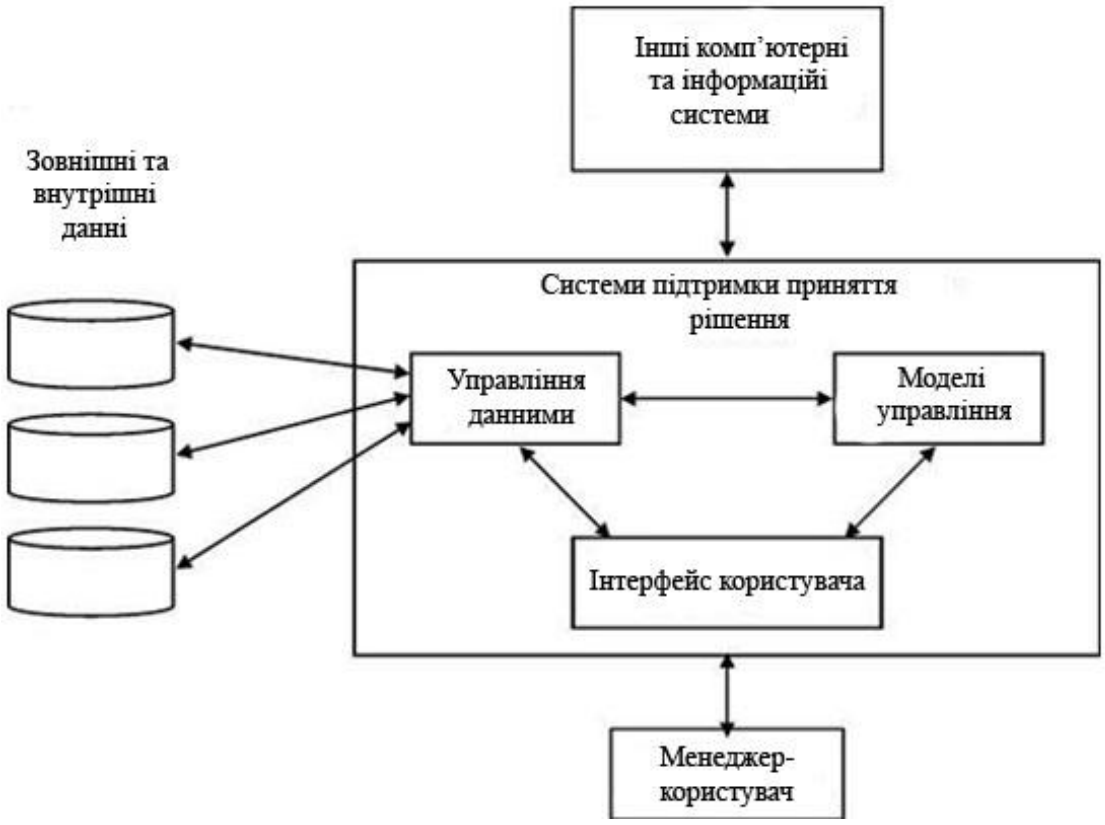


Рисунок 2.1 – Основні компоненти системи підтримки прийняття рішень

2.2 Вибір програмних технологій для розробки проекту

Таблиця 2.1 – Характеристика мов програмування та програмних технологій

Характеристика	Мови програмування		
	Java	C#	Python
Кросплатформеність	+	-	+
Середовище виконання	JDK та JRE, оновлюється спільнотою розробників	.NET та .NET Core, оновлюється розробниками компанії Microsoft	JVM, MSIL, LLVM, оновлюється спільнотою розробників
Типізація	Строга	Строга	Динамічна
Частота виходу оновлень	Висока, в середньому раз на рік	Середня, в середньому раз на 1 – 1.5 роки	Низька, раз на 2 – 2.5 роки
Технології реалізації інтерфейсу користувача	Swing, AWT	WPF, WinForms	PyQt, Tkinter
Інструменти автоматизації збірки та обробки пакетів	Ant, Maven	-	-
Бібліотеки та технології роботи с зображеннями, які підтримує мова	SciJava, SCIFIO	OpenCV	OpenCV, Scipy

Сумісність з іншими мовами програмування		F#, R	C/C++, Java
Найбільш популярні середовища розробки	IntelliJ IDEA(умовно безкоштовна), Eclipse(безкоштовна)	Rider(умовно безкоштовна), Visual Studio(умовно безкоштовна)	PyCharm(умовно безкоштовна)
Спільнота розробників	Велика, активна	Велика, не дуже активна	Середня, активна

Проаналізувавши усю наведену вище інформацію, найбільш доцільним є обрати мову програмування Java з усіма пов’язаними з нею технологіями, як тими, що відповідають вимогам до реалізації програмного продукту.

2.3 Огляд програмних технологій для розробки проекту

2.3.1 Мова програмування Java

Java — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Java вплинула на розвиток J++, що розроблялась компанією «Microsoft». Роботу над J++ було зупинено через судовий позов «Sun Microsystems», оскільки ця мова програмування була модифікацією Java. Пізніше в новій платформі «Microsoft» .NET випустили J#, щоб полегшити міграцію програмістів J++ або Java на нову платформу. З часом нова мова програмування C# стала основною мовою платформи, перейнявши багато чого з Java. J# востаннє включався в версію Microsoft Visual Studio 2005. Мова сценаріїв JavaScript має схожу із Java назву і синтаксис, але не пов'язана із Java.

Мова програмування Java - загальнооб'єктова, одночасна, строготипізована, об'єктно-орієнтована мова на основі класів. Вона зазвичай складається в наборі інструкцій байт-коду та двійкового формату, визначеного в специфікації віртуальних машин Java [8].

Переваги:

– Java пропонує більш високу роздільну здатність і портативність, оскільки програми, написані на одній платформі, можуть працювати на настільних комп'ютерах, мобільних пристроях, вбудованих системах;

– Java легко навчитися. Вона була розроблена таким чином, щоб бути простим у використанні, тому на ній легко писати, компілювати, налагоджувати та вивчати, ніж інші мови програмування.

– бібліотека класів Java дозволяє здійснювати кросплатформену розробку.

– будучи надзвичайно популярним на підприємстві, вбудованому та мережевому рівнях, Java має велику активну спільноту користувачів та підтримує доступність;

– Java є об'єктно-орієнтованою. Це дозволяє створювати модульні програми та багаторазовий код;

– на відміну від C та C ++, програми Java складаються незалежно від платформи на мові байт-коду, що дозволяє одній і тій же програмі запустити на будь-якій машині, що має встановлений JVM;

– Java має потужні інструменти розробки, такі як Eclipse SDK та NetBeans, які мають налагоджені можливості та пропонують інтегроване середовище розробки;

– продуктивна. Java існує вже понад 20 років. Весь цей час він розвивається, вдосконалюється та оптимізується.

– Сьогоднішня Java не просто відповідає продуктивності інших сучасних мов програмування, таких як Ruby або JavaScript, навіть перевершує їх у швидкості. Java також ідеально підтримує багатопроцесорні системи і дозволяє максимально використовувати свої ресурси.

– безпечна, оскільки програма, написана на Java, працює всередині віртуальної машини, вона виділяється з операційної системи. Жоден зовнішній процес не може отримати доступ до даних програми, якщо віртуальна машина не дозволяє це робити. До речі, параметри безпеки JVM можуть бути гнучко налаштовані, тому межі того, що дозволяється, перебувають під нашим повним контролем;

– Java є надзвичайно надійною мовою. Ця надійність обумовлена здатністю Java виконувати ранні та ретельні перевірки на всі можливі помилки. Компілятори Java мають можливість виявляти цілий ряд проблем, які під час роботи з більшістю інших мов з'являться лише під час виконання, це головним чином завдяки сильному розподіленню пам'яті Java та здатності автоматичного збирання сміття;

– Java багатопотокова, тобто має можливість виконувати кілька завдань одночасно в рамках програми. У Java, багатопоточне програмування було плавно інтегроване в нього, тоді як в інших мовах необхідно запускати процедури, призначені для операційної системи, щоб забезпечити можливість багатопоточності;

– відносно сумісна сумісність з однієї версії до наступної [9].

2.3.2 Apache Maven

Maven - це інструмент, який тепер можна використовувати для створення та управління будь-яким проектом на базі Java. Основна мета Maven полягає в тому, щоб дозволити розробнику зрозуміти повний стан збірки у галузі розробки в найкоротший термін. Для досягнення цієї мети існує кілька проблемних сфер, які Maven намагається спростити та виправити:

– простота процесу побудови;

– забезпечення єдиної системи збірки;

– надання якісної інформації про проект;

– надання рекомендацій щодо розробки найкращих практик;

– дозвіл прозорої міграції на нові функції.

Переваги та ключові особливості Maven:

– проста установка;

– покращений менеджмент залежностей, включаючи автома-

тичне оновлення, закриття залежностей (також відомі як перехідні залежностей);

– вміє легко працювати з кількома проектами одночасно;

– великий і зростаючий сховище бібліотек та метаданих для використання з коробки, а також механізми на місці з найбільшими проектами з відкритим вихідним кодом для отримання в режимі реального часу своїх останніх випусків;

– розширюваний, з можливістю легко створювати плагіни в Java або мовах скриптів;

– миттєвий доступ до нових функцій з невеликою або без додаткової конфігурації; моделі на основі побудови: Maven може побудувати будь-яку кількість проектів в заздалегідь визначені типи виводів, такі як JAR, WAR або розподіл на основі метаданих про проект, без необхідності робити сценарії в більшості випадків;

– послідовний сайт інформації про проект. Використовуючи ті самі метадані, що і для процесу побудови, Maven може створювати веб- сайт або PDF, включаючи будь-яку документацію, яку ви додаєте, і додає до стандартних звітів про стан розвитку проекту;

– редагування публікацій з управління та розповсюдження. Без особливої додаткової конфігурації Maven буде інтегруватися з вашою системою управління початковими ресурсами (наприклад, Subversion або Git) та керувати випуском проекту на основі певного тегу. Він також може опублікувати його в місці поширення для використання іншими проектами;

– Maven вміє публікувати окремі виходи, такі як JAR, архів, що включає інші залежності та документацію, або як джерело розподілу;

– управління залежністю: Maven заохочує використовувати центральне сховище JAR та інших залежностей. Maven постачається з механізмом, який можуть використовувати для завантаження будь-яких JAR, необхідних для побудови вашого проекту, з центрального сховища JAR, як і

CPAN Perl. Це дозволяє користувачам Maven повторно використовувати JAR у різних проектах і заохочує спілкування між проектами, щоб забезпечити вирішення проблем зворотної сумісності [10].

На рисунку 2.2 наведений приклад організації Maven.

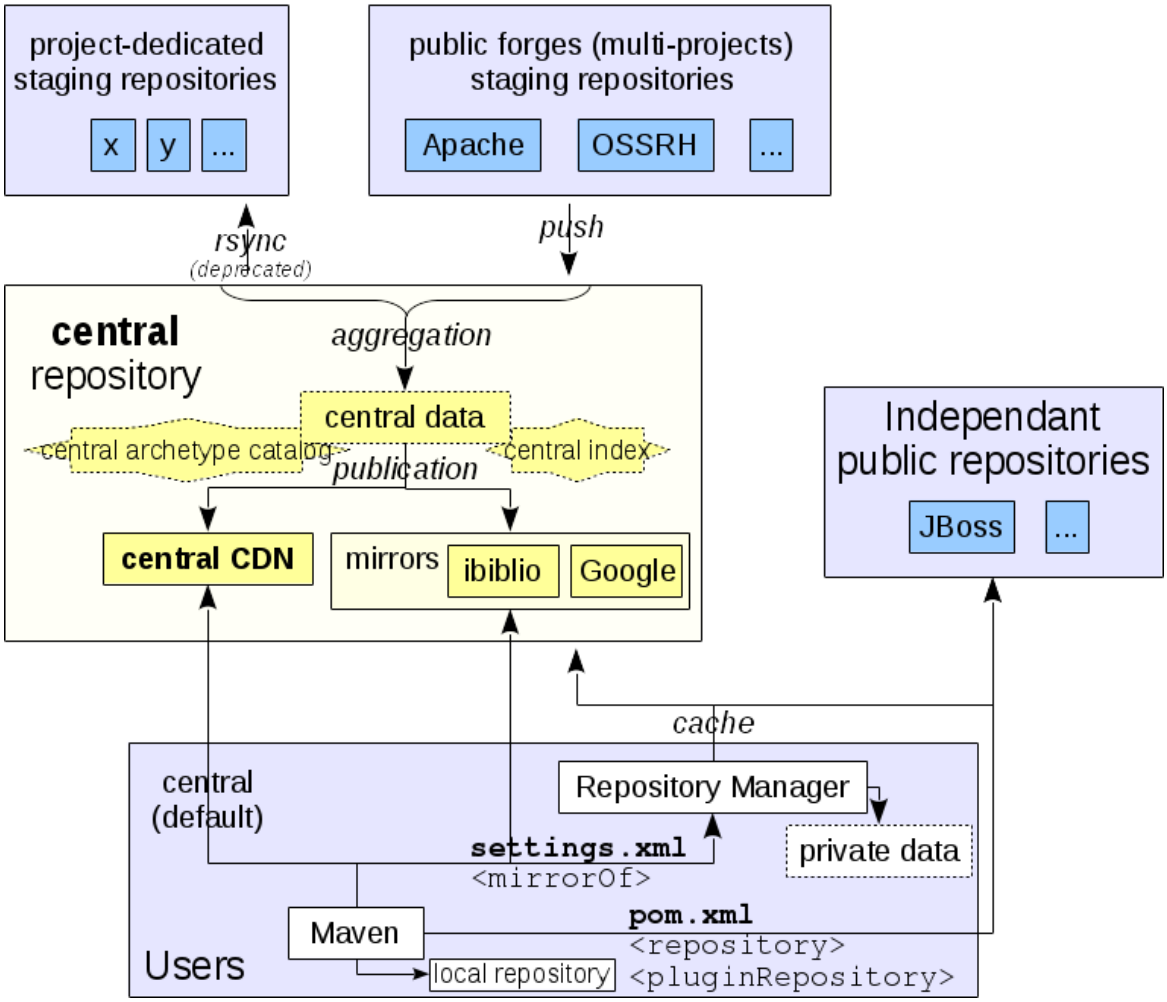


Рисунок 2.2 – Центральний репозиторій Maven

2.3.3 Фреймворк SCIFIO

SCIFIO - це гнучкий фреймворк для введення та виведення серійного формату зображення. Іншими словами, це бібліотека для читання та запису N-образних даних зображення, наприклад, до та з файлів на диску. SCIFIO побудований на бібліотеці SciJava Common. SCIFIO реалізовано як набір плагінів для плагіна SciJava. Його ядро написано під дозволеною ліцензією

BSD, щоб максимізувати свободу включення як відкритих, так і закритих програм. Рада SciJava збирає плагіни в контексті додатків, які, як правило, доступні через Служби. Таким чином, SCIFIO визначає набір плагінів та служб, що полегшують введення / виводу зображення. Розробники, як правило, починають з самого класу SCIFIO: Шлюз до контексту SciJava, що забезпечує зручні методи доступу для функціональних компонентів SCIFIO.

Система SciJava сортує плагіни за допомогою типу "type", що представляє роль даного плагіна. Розширюваність і гнучкість досягаються за допомогою надання публічного сервісного API, який організовує та делегує доступні плагіни кожного типу. Таким чином, розробка SCIFIO стосується, перш за все, додавання нових реалізацій плагіна для досягнення бажаного результату. У наступних розділах описуються основні типи плагінів SCIFIO та поведінка, яку вони контролюють.

Перш за все це Формат. Формати - це сукупність компонентів, керованих інтерфейсом (рисунок 2.3), які визначають етапи декодування джерела зображення до його метаданих та значень пікселів. Формат завжди повинен містити компонент метаданих, який визначає його унікальні поля та структури, такі як деталі інструмента для отримання, типи вісь розміру або довжини хвиль випромінювання детектора. Кожна реалізація метаданих повинна також бути в змозі виразити себе як стандартний об'єкт ImageMetadata, що не залежить від формату, встановлюючи загальну базу для використання в рамках.

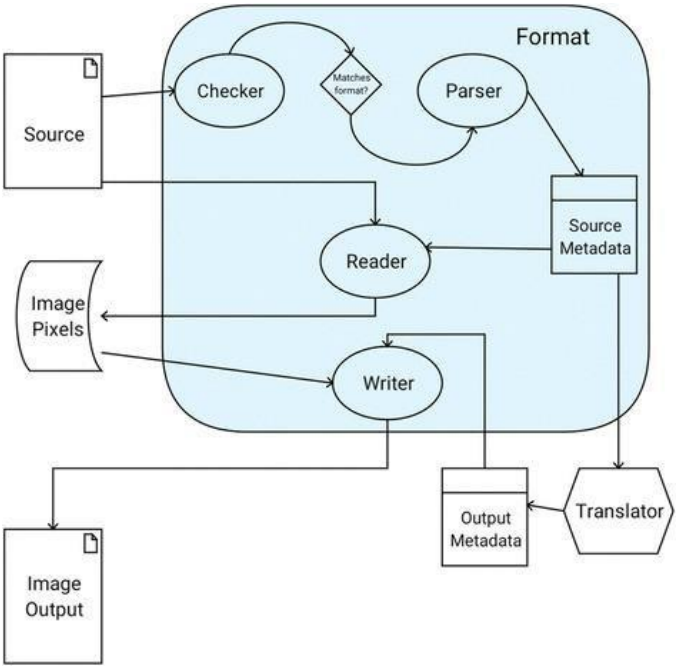


Рисунок 2.3 – Компоненти плагіну Format та їх роль у вводі-виводі зображення

Компонент Checker містить логіку для відповідності певного формату з потенційним джерелом зображення, тоді як компонент Parser виконує фактичне створення метаданих з цього джерела. Компоненти Reader і Writer використовують метадані для читання та запису даних пікселів, відповідно. З огляду на мету вільно обмінюваних даних зображення, Writers є необов'язковими компонентами, і їх не слід застосовувати для власних форматів.

Другим важливим типом плагіна є Translator, який кодує логіку для перетворення з одного типу метаданих на інший. Перекладачі дозволяють стандартизувати фірмові формати для загальних структур метаданих, таких як OME, і, отже, відігравати ключову роль у перетворенні зображень між форматами. Перекладачі, як правило, створюються для супроводу письменників, забезпечуючи належне заповнення метаданих формату. Крім того, система Translator дозволяє інтегрувати нові формати відкритого обміну за допомогою бібліотек, що підтримують лише перекладачі, і конвертувати підтримувані типи метаданих до нового стандарту. Приклад цієї моделі можна побачити у компоненті SCIFIO-OME-XML (рисунок 2.4).

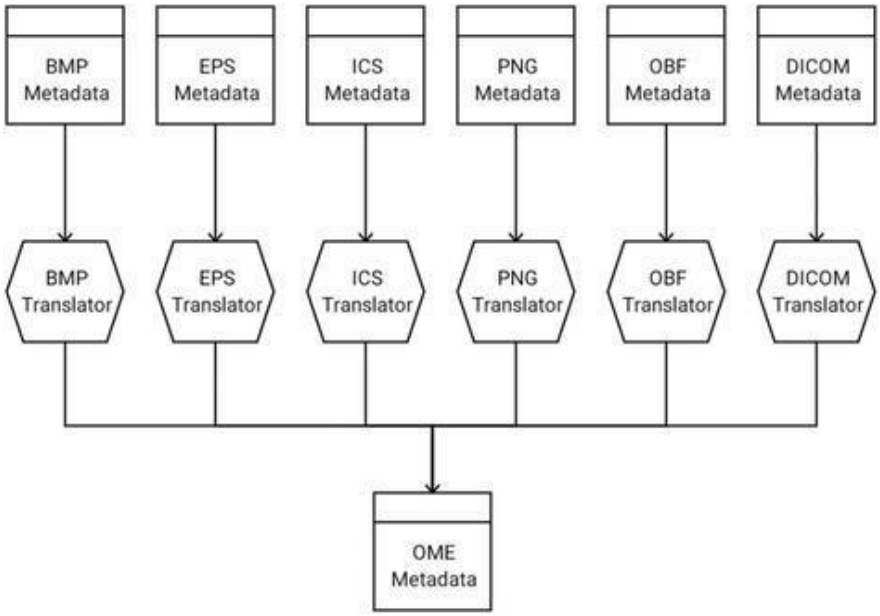


Рисунок 2.4 – Набір SCIFIO-OME-XML Translator, для перетворення метаданих у формат відкритого обміну OME-TIFF

Хоча Format та Translator додають нову поведінку до базової структури, SCIFIO також має типи плагінів для керування існуючою поведінкою. Наприклад, фільтри плагінів надають механізм форматування-агностик для зміни поведінки Reader. Фільтри створюють замовлений ланцюжок делегування, кожен з яких працює за даними свого батька, і може бути індивідуально включений "на" або "вимкнено" на основі кожного читача. Зразок поведінки укладання фільтра ілюструється у ChannelFiller для перетворення пікселів "індексованих кольорів" у значення RGB та FileStitcher для об'єднання декількох файлів на диску для створення одного набору даних (рисунок 2.5 – Поведінка плагінів ChannelFiller та FileStitcher Filter).

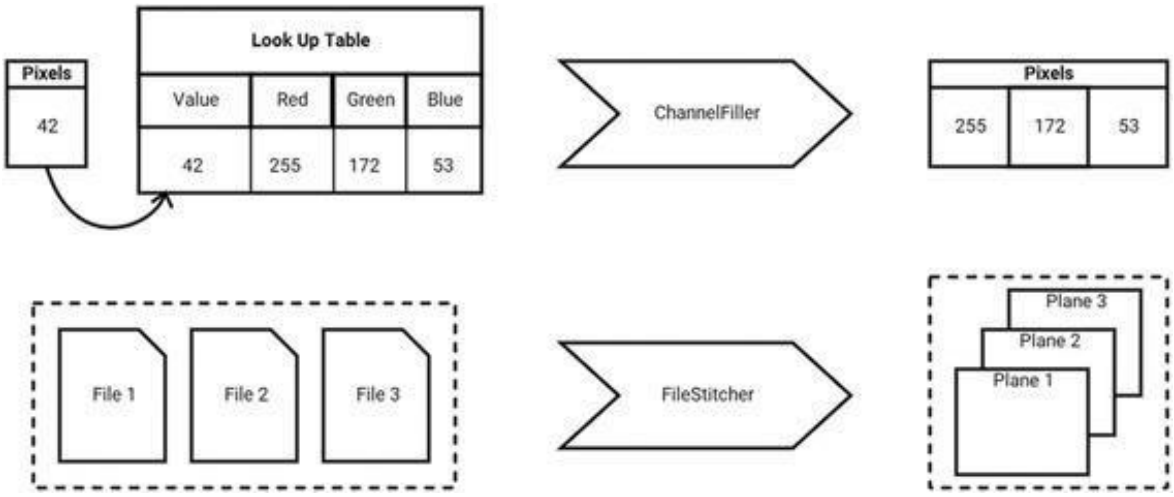


Рисунок 2.5 – Поведінка плагінів ChannelFiller та FileStitcher Filter

За допомогою усіх плагінів SciJava числове значення пріоритету, яке додається до кожного класу, створює неявне відносне замовлення для операцій, наприклад, порядок запитів Checker, запит перекладачів або фільтр. Пріоритети автоматично враховуються при використанні служб SCIFIO: від компонент Checker для опитування FormatService доTranslatorService, який знаходить правильний перекладач для даного запиту, пріоритети дозволяють спочатку запитувати найбільш конкретні рішення, перш ніж перейти до більш загальних параметрів. Ці частини разом забезпечують надійну та гнучку бібліотеку для читання та запису даних зображень [11].

2.3.4 Swing API

Swing API являє собою набіррозширюваних компонентів графічного інтерфейсу для полегшення життя розробника для створення Java- додатків на базі Front End / GUI. Він побудований на вершині API AWT і діє як заміна AWT API, оскільки він має майже кожен елемент керування, що відповідає контролю AWT. Компонент Swing слід за структурою Model- View-Controller

для виконання наступних критеріїв:

– єдиний API повинен бути достатнім для підтримки декількох видів і відчуттів.

– API повинна бути моделлю, так що для API найвищого рівня не потрібно мати дані.

– API - це використовувати модель Java Bean, щоб інструменти Builder і IDE могли надавати кращі послуги розробникам для використання.

Архітектура Swing API слідує за вільно заснованою архітектурою MVC наступним чином:

– модель представляє дані компонента;

– вид - це візуальне представлення даних компонента;

контролер приймає вхід від користувача у вікні перегляду та відображає зміни в даних компонента;

– Swing компонент має модель як окремий елемент, а частина перегляду та контролера вкладена в елементи інтерфейсу користувача. Через це Swing має висувну архітектуру [12]. На рисунку 2.6 показана архітектура Swing MVC додатку.

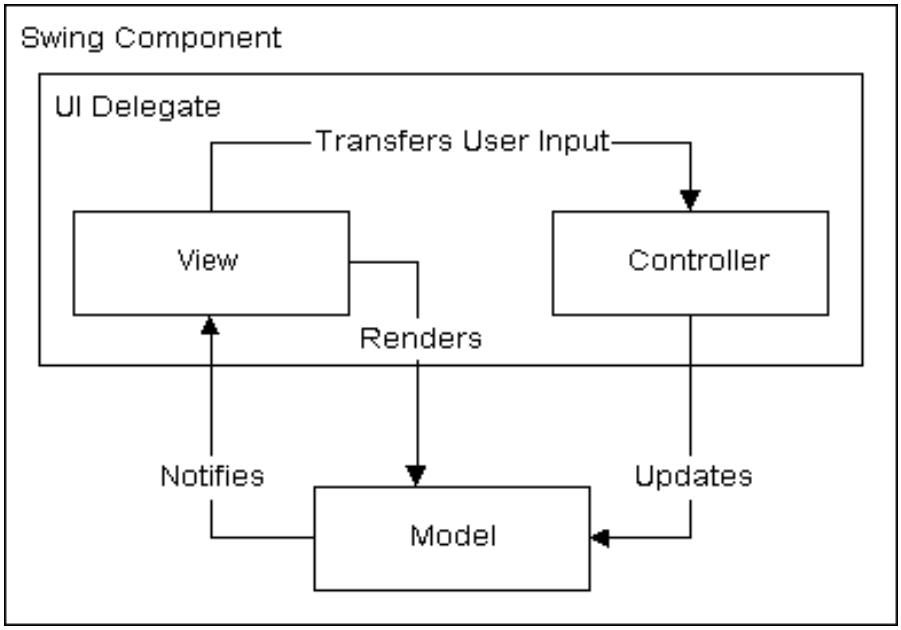


Рисунок 2.6 – Реалізація підходу MVC у Swing

Swing надає багато нових можливостей для тих, хто планує писати великомасштабні програми на Java. Ось деякі з найбільш популярних функцій:

– привабливий вигляд і відчуття. Функція відключення зовнішнього вигляду, що зникає, дозволяє нам пристосувати зовнішній вигляд програми та аплетів до стандартного вигляду, як Windows, так і Motif. Ми можемо навіть перейти на різний вигляд і відчувати себе під час роботи. Swing має можливість підтримувати кілька поглядів і відчуває, але в даний час вона надає підтримку для Windows і Motif (рисунок 2.7 – Вигляд Motif Look and Feel). Оскільки зовнішній вигляд компонентів контролюється Swing, а не операційною системою, відчуття компонентів також може бути змінено. Вигляд і відчуття компонента можна розділити з логіки компонента. Таким чином, можна "підключити" новий вигляд і відчувати себе для будь-якого компонента, не впливаючи на решту частини коду;

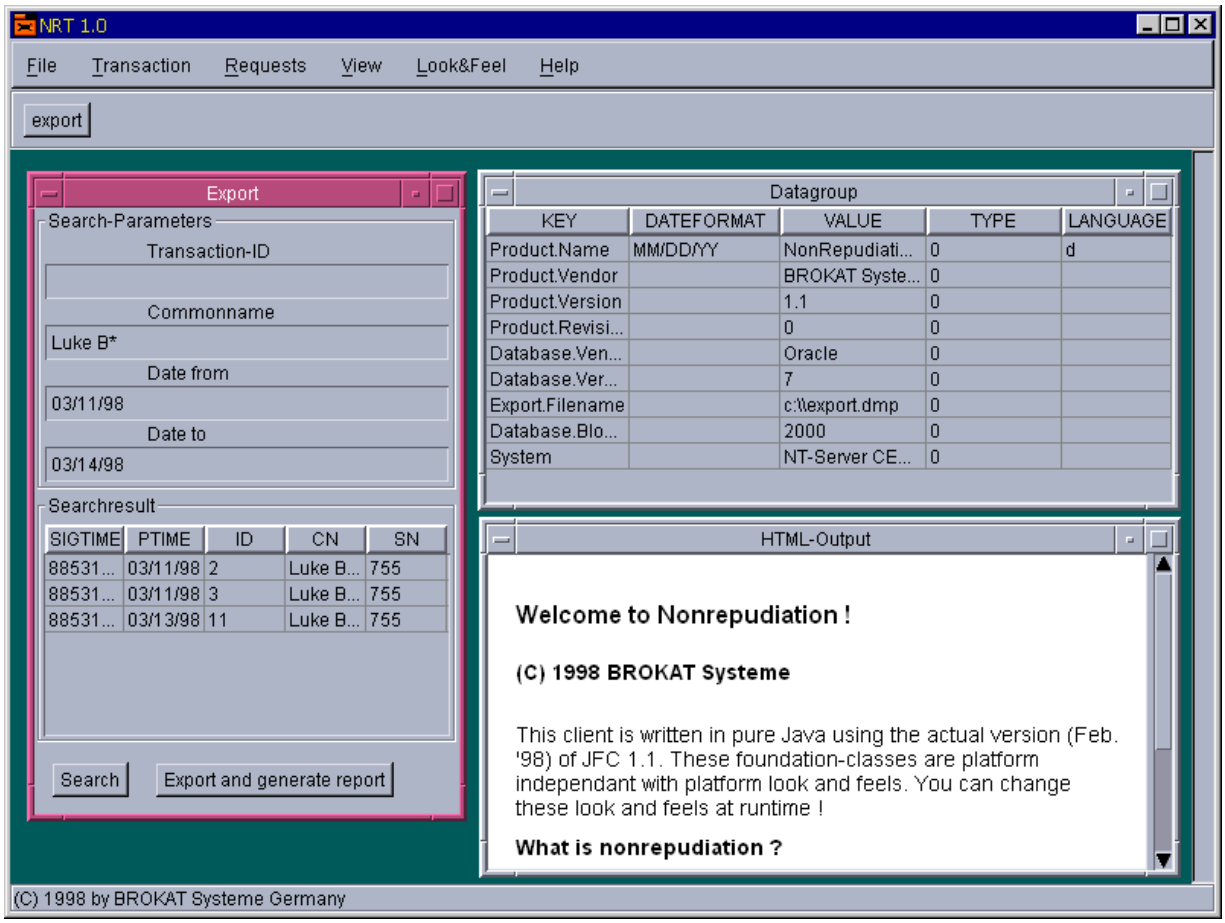


Рисунок 2.7 – Вигляд Motif Look and Feel

Більшість Swing компонентів є легковажними. Вони використовують спрощені графічні примітиви, щоб малювати на екрані, і навіть можуть дозволити фрагменти бути прозорими. З легкими компонентами кожен компонент перетворює себе з використанням примітивних рисунків графічного об'єкта (наприклад, drawLine (), fillRect () та ін.). Легкі компоненти завжди роблять себе на поверхні суперважкої компонентів верхнього рівня, які вони містять. З приходом JDK 1.1 програмісти можуть безпосередньо розширювати класи java.awt.Component або java.awt.Con-tainer при створенні легких компонентів . На відміну від java.awt.Canvas або java.awt.Panel, ці класи не залежать від рідного peer і дозволяють розробнику швидко перейти до графічного контексту контейнера. Це призводить до швидшого, меншого обсягу пам'яті компонентів, ніж раніше було доступно в Java. Майже всі компоненти Swing є легковажними; лише декілька контейнерів верхнього рівня є важкими. Ця конструкція дозволяє програмістам малювати (і перемальовувати) зовнішній вигляд своєї програми під час виконання, а не зв'язувати його з Look and Feel операційної системи хосту. Крім того, дизайн компонентів Swing підтримує легку модифікацію поведінки компонентів. Наприклад, ви можете вказати майже будь-який компонент Swing, чи бажаєте ви його приймати або відхилити фокус, і як він повинен обробляти введення на клавіатурі [13].

Кілька інших функцій відрізняють Swing від старших компонентів AWT:

– широкий вибір нових компонентів, таких як таблиці, дерева, слайдера, прогалини, внутрішні кадри та текстові компоненти;

– компоненти Swing містять підтримку для заміни своїх вставок довільною кількістю концентричних кордонів;

– компоненти Swing можуть мати підказки, розташовані над ними. Підказка - це текстовий спливаюче вікно, яке миттєво з'являється, коли курсор миші знаходиться в області живопису компонента. Підказки можна ви-

користовувати для отримання додаткової інформації про відповідний компонент;

– ви можете довільно прив'язувати події клавіатури до компонентів, визначаючи, як вони будуть реагувати на різні натискання клавіш у заданих умовах;

– існує додаткова підтримка налагодження для відтворення ваших власних легких компонентів Swing.

2.3.5.1 SciJava Common

SciJava Common - це загальна бібліотека для програм SciJava. Він забезпечує плагіни рамки, з розширюваним механізмом для виявлення служби, за підтримки власного процесора анотації, так що плагіни можуть завантажуватися динамічно. У першу чергу, SciJava Common - це плагіновий каркас - основа для розробки модульно-розширюваних Java-додатків [14].

Ось декілька основних основних служб SciJava Common:

– AppService - відстежує програмне забезпечення, що містяться у контексті;

– DisplayService – відстежує доступні дисплеї, а також активний дисплей, і надає засоби для створення нових дисплеїв для візуалізації даних;

– EventService - публікує події на автобусі події та дозволяє зацікавленим сторонам підписатися на них. Служба забезпечує центральний спосіб зв'язку між різними частинами коду бази;

– IOService – загальні інструменти для відкриття та збереження даних у контексті;

– MenuService – Створює структуру меню програми;

– ThreadService – керує багатопоточуванням;

– ModuleService – відстежує доступні модулі та надає інфраструктуру для їх виконання;

– ObjectService – відстежує доступні об'єкти різних типів, включаючи набори даних та дисплеї;

– OptionsService – Інструменти для керування налаштуваннями програми;

– PlatformService – Надає гачки для розширення поведінки програми залежно від платформи розгортання (операційна система, версія Java та ін.);

– PluginService – відстежує доступні плагіни та надає інфраструктуру для їх виконання (за допомогою модуля);

– StatusService – публікує оновлення статусу поточних операцій;

– ThreadService – керує багатопоточуванням;

– ToolService – Відстежує доступні інструменти - логіку, що зв'язує вхід користувача до поведінки, а також активний інструмент (вибраний на панелі інструментів);

– UIService – виявляє та запускає користувальницький інтерфейс.

2.3.5.2 ImgLib2

ImgLib2 - універсальна, багатовимірна бібліотека для обробки зображень. Він забезпечує інтерфейсно-керований дизайн, який підтримує числові та нечислові типи даних (8-бітне ціле число без знаку, 32-розрядну плаваючу точку та ін.) розширюваним способом. Він реалізує декілька джерел даних та організацій зразків, включаючи один єдиний примітивний масив, один масив на площину, N-мірний масив "клітинки", кешовані до диска та за запитом, та літаки, читаються за запитом з диска.

Основний дизайн ImgLib2 базується на трьох основних поняттях: Accessibles (тобто зображення), Accessors і Types. Ми визначаємо зображення як будь-яке відображення від підмножини n-мірного евклидова координатного простору до загального типу пікселів. Властивості зображення виражаються в доступних інтерфейсах: координати можуть бути якцілими, так і реальними, домен координувати може бути обмежений або нескінченним, зображення може підтримувати довільний доступ у довільних координатах та / або ітерації всіх зразків. Розглянемо звичайний образ пікселя. Він містить зразки певного типу значення в обмеженому n-мірному просторі, розташованому на цілій сітці, і є одночасно доступним випадково (у довільних цілих координатах) і ітерабельно. Важливо, що ImgLib2 підтримує концепції, що виходять за рамки звичайного піксельного зображення, наприклад нескінченні, процедурно згенеровані зображення або безперервні зображення, інтерпольовані з малозабезпечених даних.

Доступ до зразків (пікселів) значень та координат забезпечується через інтерфейси Accessor. Вони існують у варіантах для цілих чи справжніх координат, а також ітераційного та випадкового доступу. Для ітерації додатків ітераційне замовлення підлягає впровадженню, що спеціалізується на кожній схемі пам'яті, щоб мінімізувати час доступу.

Accessors надають доступ до вартості через типи. ImgLib2 має ієрархію інтерфейсів Type, які описують алгебраїчні властивості сімейств конкретних типів. Приклади - порівняльні типи чи NumericTypes, які підтримують основні арифметичні операції (+, -, *, /).

Шаблони доступу та властивості типу дозволяють дрібнозернисту специфікацію алгоритмічних вимог. Алгоритм, який створюється з

використанням відповідних інтерфейсів, застосовується до будь-якого конкретного зображення, що реалізує ці інтерфейси. Повторне використання алгоритмів максимізується шляхом визначення їх для мінімального набору необхідних властивостей.

У Java цей рівень загальності вимагає, щоб пікселі були об'єктами.

Зберігання простих значень пікселів (наприклад, байтів) як окремих об'єктів, однак, поставляється з значними запасами пам'яті. І навпаки, створення нових об'єктів на піксельний доступ вводить значні накладні витрати на виконання і запускає частому збирання сміття. Обидва підходи не добре масштабуються з великими зображеннями. Щоб вирішити це питання, ImgLib2 використовує типи проксі для доступу до даних пікселів, які можуть бути віднесені до масивів примітивних типів Java (байт, поплавок та ін.). Таким чином, прилад може повторно використовувати один проксі-екземпляр для всіх піксельних звернень. У наведеному вище прикладі проксі тип T інвенціонує один раз, а потім повторно використовується в кожній ітерації, змінюючи лише внутрішній стан [15]. Цей віртуалізаційний шаблон не має додаткових витрат у порівнянні з прямим доступом до масиву, завдяки оптимізації, проведеному компілятором Java у режимі реального часу (JIT). На рисунку, що знаходиться нижче (рисунок 2.8 – Знаходження та візуалізація локального мінімуму після Гаусівського зашумлення) зображено результат роботи гаусівського модуля бібліотеки ImgLib2 на практиці. Ми чітко можемо розуміти як працює данна програма та який функціонал в ній присутній, тобто для нас не буде несподіванкою якщо щось трапиться в програмі, це легко буде відформатувати або заново розробити при наявності базових навичок.

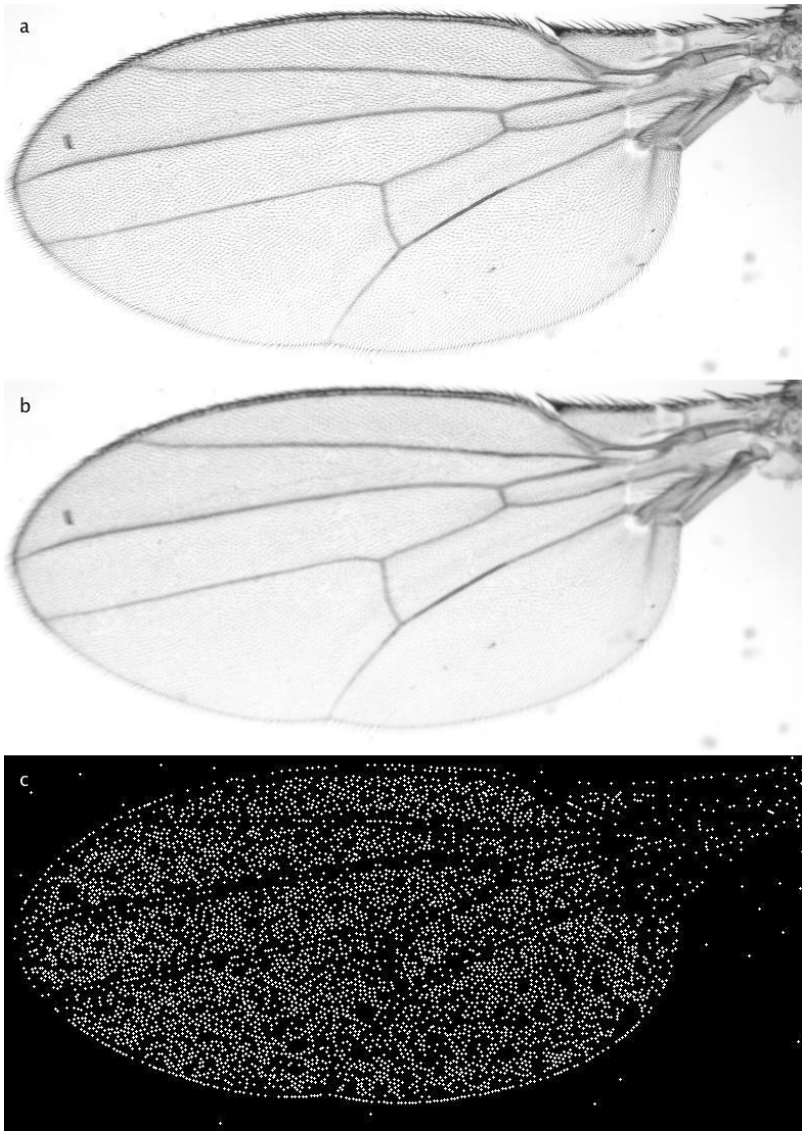


Рисунок 2.8 – Знаходження та візуалізація локального мінімуму після Гаусівського зашумлення

Висновки до розділу 2

В даному розділі було виконано огляд технологій, які використовуються для розробки даного програмного продукту, приведені їх основні концепції, можливості, переваги та роль у побудові дипломного проекту. Також були детально проаналізовані і досліджені всі переваги та недоліки кожної з технологій, обрано доцільний їх набір. Досягнуте повне розуміння того, яка технологія за що відповідає і яку роль грає у створенні проекту.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Структура побудови програми

При побудови додатку використовувалась модульна архітектура, що дозволяє легко масштабувати програмний продукт, полегшує роботу над ним новим розробникам, дозволяє розширити. Також, модульна структура розширює можливості версійного оновлення, бо розробка маленьких модулів проходить швидко.

Структура програми включає до себе інтерфейс користувача, точку доступу до модулів, яка реалізована за паттерном програму “Одинак”, та модулів-контроллерів, які включають до себе всю логіку програми (рисунок 3.1).

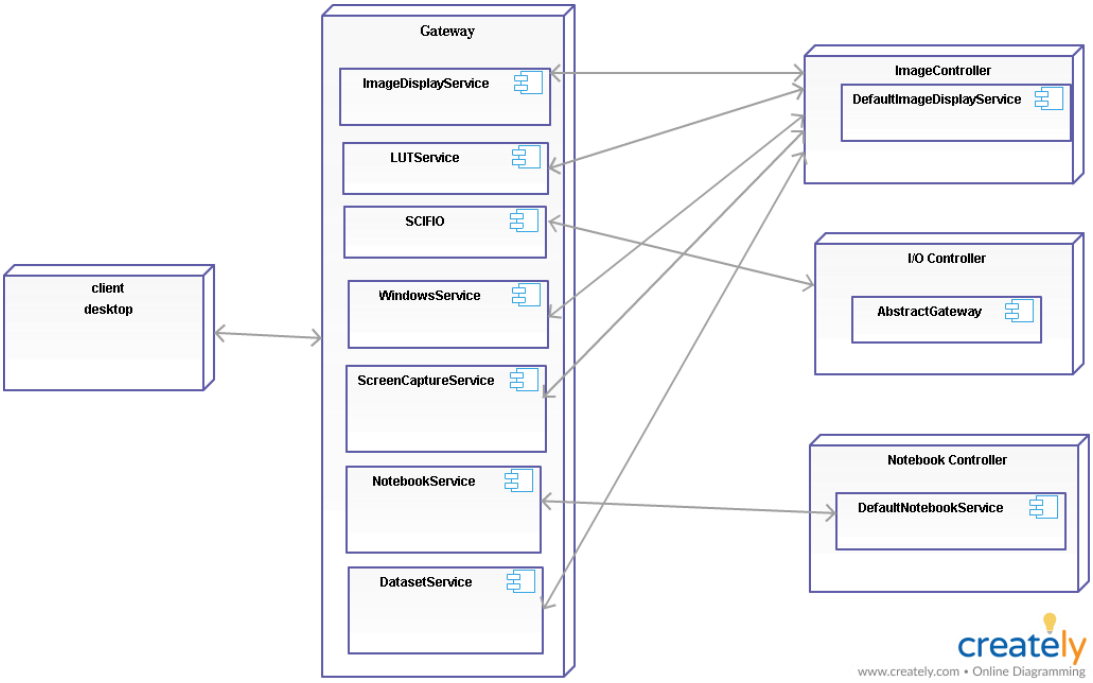


Рисунок 3.1 – Структурна діаграма програмного продукту

3.2 Додатки, використані при розробці проекту

Для оптимізації розробки проекту, а також для зручності, були використані наступні додатки:

– редактор програмного коду IDEA;

– розподілена система керування версіями файлів Git;

– Sublime Text для аналізу та перевірки даних, що зберігаються в файлах конфігурації;

– інструмент BIRT для конфігурування файлів .roi;

– Launch4j для створення файлів .exe.

Кожен з цих додатків володіє рядом унікальних, незамінних властивостей, полегшуючих розробку і тестування програмних додатків.

3.2.1 Редактор програмного коду IDEA

В інтернеті та серед користувачів найбільшої популярності набув редактор програмного коду для Java – IntelliJ IDEA. Так само його вже почали згадувати в книгах. Це платний проект, але можна скористатися безкоштовною пробною версією і зрозуміти всі його переваги та недоліки. Скачати можна з сайту https://www.jetbrains.com/idea/

Середовище розробки IntelliJ IDEA підтримує і Mac, і Windows, і (рисунок 3.3). Взагалі редактор універсальний за своїм призначенням.

Можливості редактора:

– інтегроване модульнетестування;

– перевірки коду;

– інтегрований контроль версій;

– інструменти рефакторінга коду;

– набір інструментів для навігації проекту;

– виділення і автоматичне завершення;

– підтримка Maven, Gradle, Ant, Gant, SBT, NPM, Webpack, Grunt, Gulp та інших інструментів збірки;

– інструменти декомпіляції;

– термінал;

– підтримка Docker машин;

– підтримка JVM та non-JVM фреймворків;

– інструменти роботи з базами даних;

– можливість зміни інтерпретатора в один натиск;

– підтримка основних програмних серверів: Tomcat, JBoss,

– автоматичне імпортування бібліотек [16].

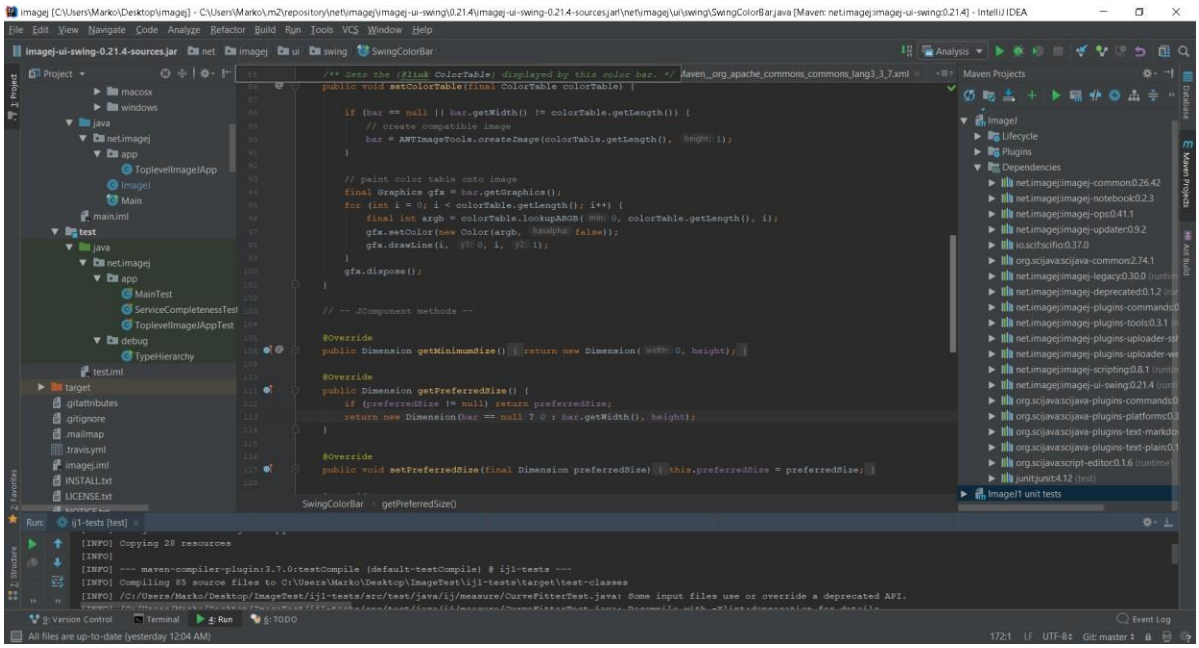


Рисунок 3.3 – Середовище розробки «Intellij IDEA»

3.2.2 Sublime Text

Sublime Text – це текстовий редактор, поширюваний за платною підпискою, написаний на C ++, який:

– кросплатформений (працює в Linux, OS X і Windows);

– легковісний та швидкість у роботі;

– має зручний інтерфейс (включаючи всілякі анімації) (Рисунок– Інтерфейс «SublimeText»);

– гнучко настроюється;

– має велику бібліотеку розширень-плагінів;

– підтримує VIM-режим.

Редактор умовно-безкоштовний або умовно-платний. Можливо сплатити 80$ за ліцензію, але їм можна користуватися безкоштовно.

В безкоштовному режимі елементи інтерфейсу періодично пропонують сплатити за ліцензію. Це взагалі не робить розробнику ніяких труднощів, бо функціонал безкоштовної версії не відрізняється від платної, тому користуватися ним може будь-який охочий [20].

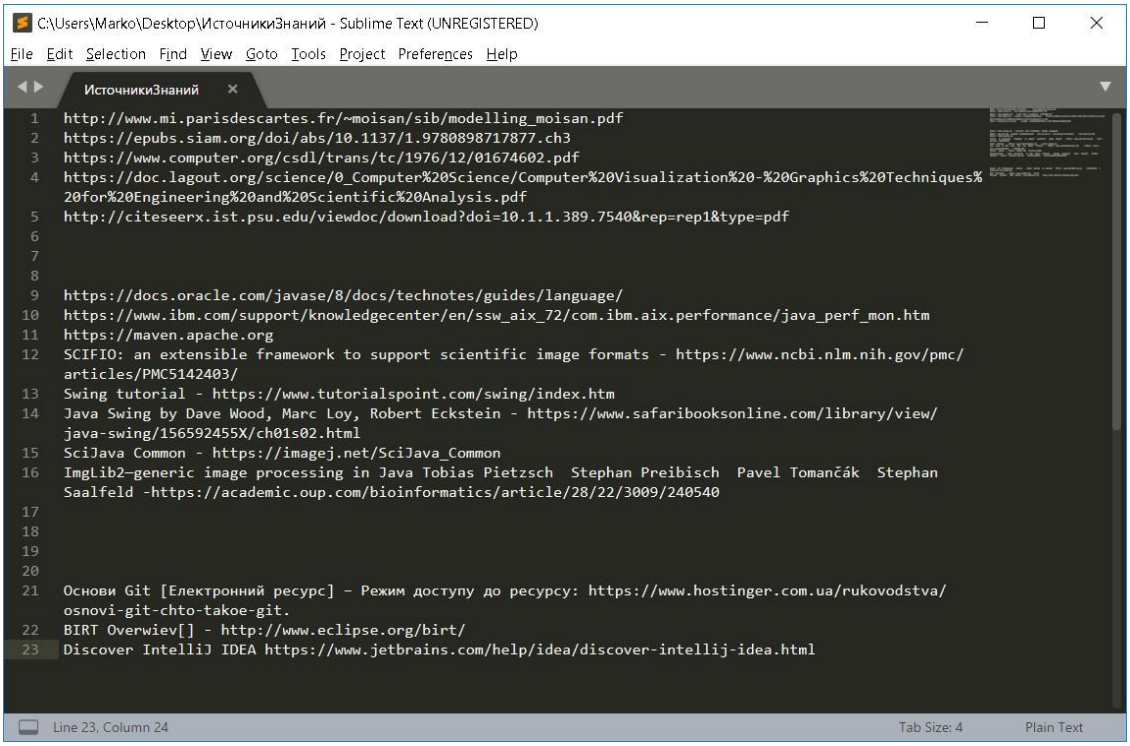


Рисунок 3.4 – Інтерфейс «Sublime Text»

Також є можливість встановити плагін для роботи з xlm (xlm plugin). Цей плагін включає можливість ручного форматування xml файлу за вашим бажанням та цілями [21].

3.2.3 Розподілена система керування версіями файлів Git

Система контролю версій або VCS допомагає організувати процесс розробки для розробників, які мають потребу у аналізі змін і відстеженні прогресу розробки іншими розробниками в команді. Очевидно, що система контролю версій – один із важливих механізмів в розробці програмного забезпечення. VCS дають можливість призначати для певних змін / ревізій / оновлень літерні або числові значення. Кожен розробник

має унікальний ідентифікатор, який дозволяє відстежувати прогрес работи, зміни коду та вклад, який він вніс до проекту [18].

Git – повністю безкоштовний програмний продукт, який має веб- платформу та локальний GUI, є кросплатформеним, тобто працює на усіх основних операційних системах.

Функції Git:

– розподілена система управління версіями, Git дотримується принципу тимчасової мережі - peer to peer (рівний до рівного) на відміну від інших систем на кшталт Subversion (SVN), яка заснована на моделі client-server (клієнт-сервер);

– Git дозволяє розробникам мати безліч абсолютно незалежних гілок коду. Створення, видалення та об'єднання цих гілок відбувається без будь-яких проблем і великих витрат часу;

– у Git всі операції атомарні; це означає, що будь-яка дія може бути повністю вдалим або провалитися (без будь-яких змін). Це дійсно важливо, так як в деяких системах контролю версій (на кшталт CVS), де дії не атомарний, деякі операції, які зависли, можуть залишити його в нестабільному стані;

– на відміну від інших VCS, таких як SVN або CVS, де метадані

зберігаються в прихованих папках (.cvs, .svn, і т.д.), в Git всі дані розташовані в каталогах .git;

– він використовує модель даних, яка допомагає забезпечити

криптографічний цілісність всього, що є присутнім в репозиторії. Кожен раз коли файли додаються або робиться комміт (commit), генеруються їх контрольні суми; аналогічний процес відбувається при їх витяганні. Розробники в результаті мають декілька версій файлів та можуть повертатися до них. (рисунок 3.5 – Схема розподіленої системи контролю версіями файлів);

– ще одна чудова функція, яка присутня в GIT – це його індекс. В межах індексу, розробники можуть форматувати комміти і переглядати їх до фактичного застосування [17].

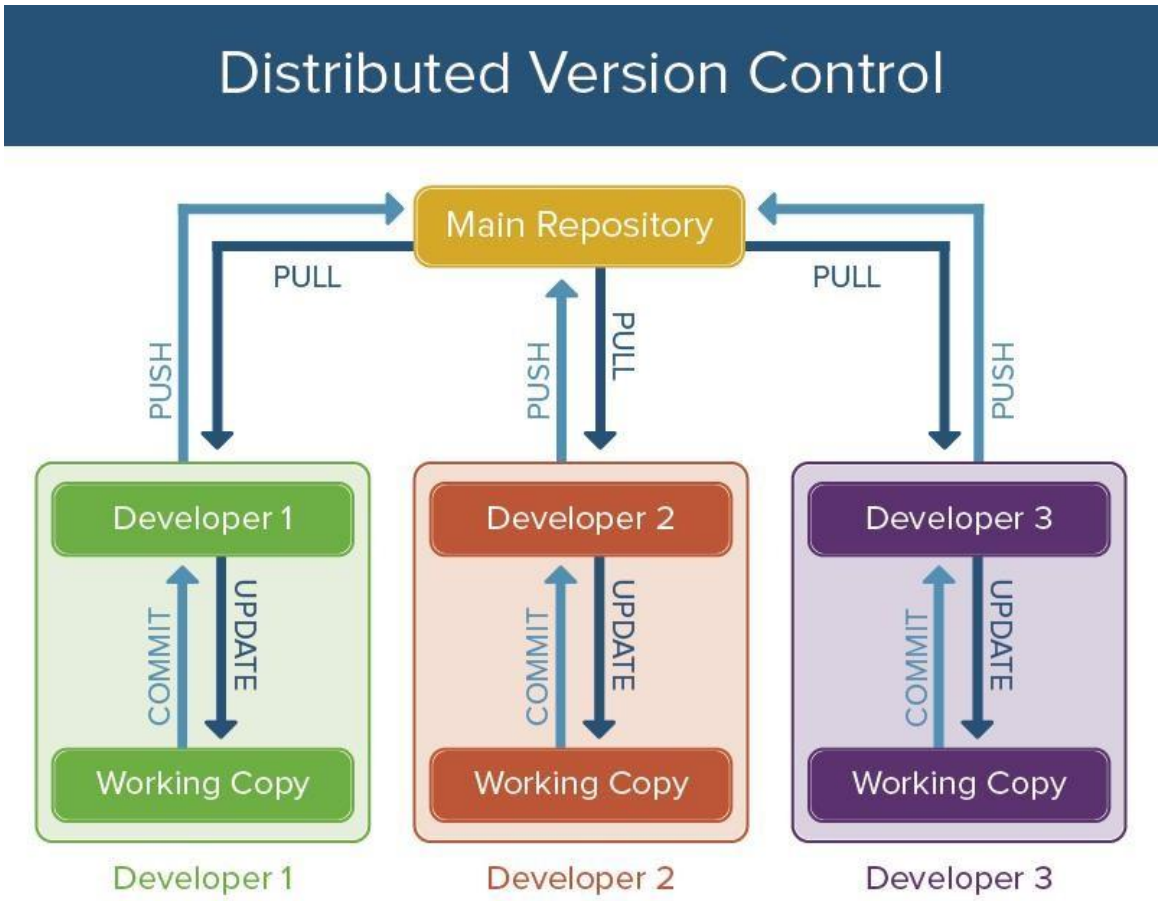


Рисунок 3.5 – Схема розподіленої системи контролю версіями файлів

3.2.4 BIRT

BIRT - це проект із відкритим вихідним кодом, який надає технологічну платформу BIRT для створення візуалізації даних та звітів, які можна вбудувати в багаті клієнтські та веб-додатки, особливо ті, що базуються на Java та Java EE. Це безкоштовний проект. Скачати можна з сайту http://www.eclipse.org/birt/ (рисунок 3.6 – Інтернет сторінка «BIRT»).

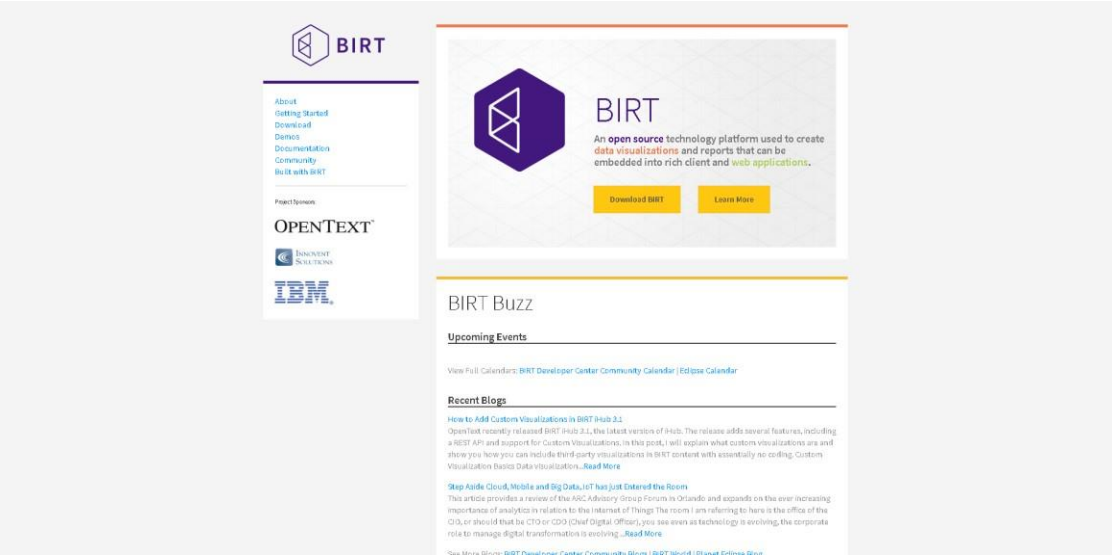


Рисунок 3.6 – Інтернет сторінка «BIRT»

BIRT має дві основні компоненти: дизайнер звітів на основі Eclipse та компонент виконання, який ви можете додати до сервера додатків. BIRT також пропонує двигун діаграм, який дозволяє додавати діаграми до власної програми. Оскільки BIRT реалізовано на мові програмування Java, він є кросплатформеним додатком, тобто підтримується Windows, Linux та Mac, оскільки вони підтримують JDK та JRE (рисунок 3.7).

Можливості програми:

– компонентна модель для повторного використання конструкцій та елементів;

– зручність використання функцій у форматі WYSIWYG перетягування та падіння;

– підтримка широкого кола звітів, макетів та форматування;

– програмний контроль, тому вміст звіту можна створювати та управляти за допомогою сценаріїв;

– доступ до даних у кількох джерелах даних [19].

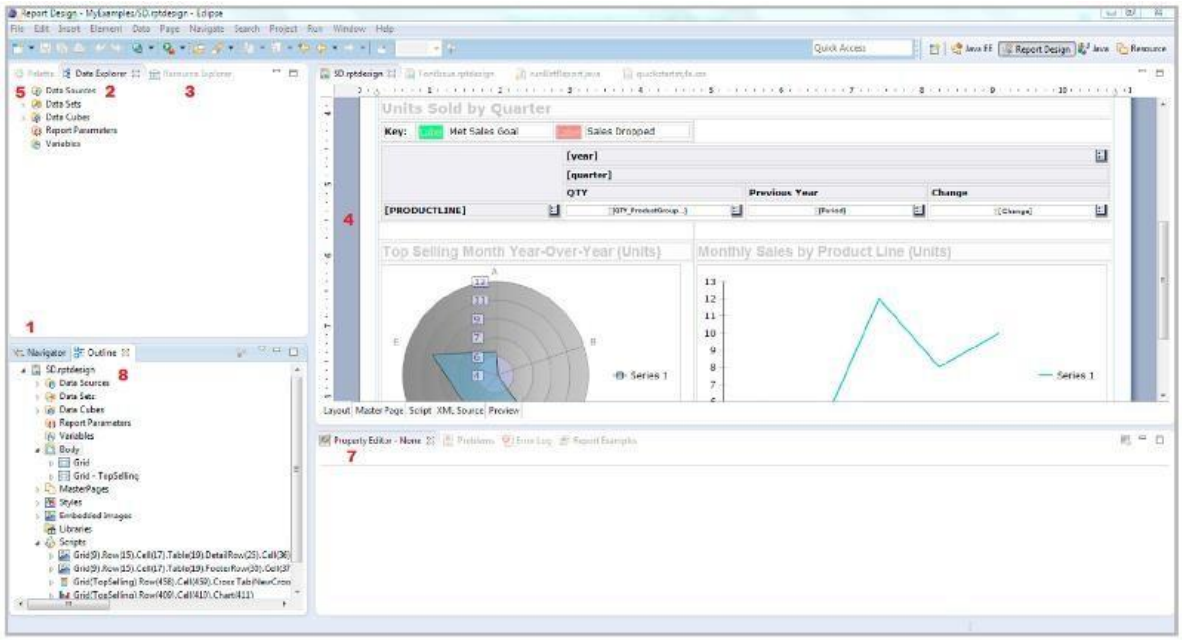


Рисунок 3.7 – Середовище роботи зі звітами «BIRT»

3.2.5 Launch4j

Launch4j - це крос-платформний інструмент для обгортання Java- додатків, що розповсюджуються як баночки в легких звичайних виконуваних файлах Windows. Виконавчий файл може бути налаштований на пошук певної

версії JRE або використовувати пакетну версію, і можна встановити параметри виконання, такі як розмір початкової / максимальної купи. Обгортка також забезпечує кращу користувальницьку роботу за допомогою піктограми програми, власного екрана попереднього виявлення JRE та сторінки завантаження Java у випадку, якщо відповідної JRE неможливо знайти. Інструмент цілком безкоштовний та може використовуватися в будь-яких цілях (рисунок 3.8). Можливості інструменту:

– обгортає банки в оригінальному виконуваному файлі Windows і дозволяє запускати їх як звичайну програму Windows. Можна загорнути програми на Windows, Linux і Mac OS X;

– створює пускові установки для банок і файлів класу без обгортання;

– підтримує виконувані бари та динамічну роздільну здатність кластеру, використовуючи змінні середовища та символи підстановки;

– працює в комплекті з JRE або шукає найновіші Sun або IBM JRE / JDK в заданій версії діапазону та типу (64-розрядний або 32- розрядний);

– підтримка графічного інтерфейсу та консольних додатків;

– пропускає аргументи командного рядка, також підтримує постійні аргументи;

– забезпечує доступ до змінних середовища, реєстр та шлях виконуваного файлу через властивості системи;

– встановлює змінні середовища;

– можливість змінити поточний каталог на виконуване місце розташування;

– підтримує особливості безпеки Windows комплекту сертифікації Windows 8;

– легковажний;

загорнута програма працює на всіх платформах Windows, Launch4j

працює на Windows, Linux і Mac OS X [22].

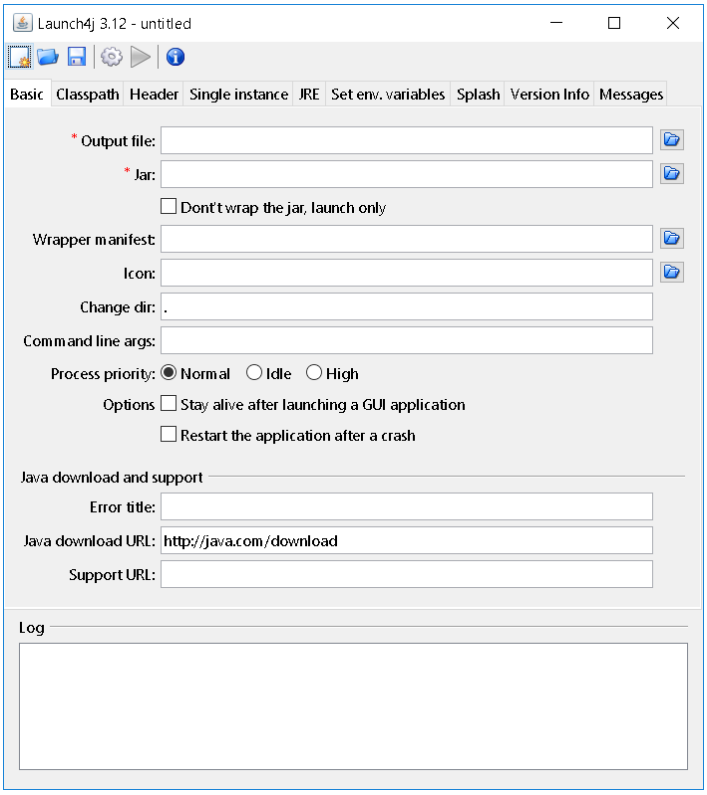


Рисунок 3.8 – Інтерфейс інструменту «Launch4j»

Висновки до розділу 3

В даному розділі були розглянуті програмні засоби, за допомогою яких створювався даний програмний продукт, особливості їх функціоналу та переваги над іншими додатками схожої області використання. Також були розглянуті додатки, за допомогою яких було автоматизовано процес розробки, організації та збірки додатку, що оптимізувало зусилля та зменшило тривалість розробки.

4. ОПИС РОЗРОБКИ ТА РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

4.1 Встановлення програмного забпечення

Для запуску програми необхідно:

- Встановити на робочу систему Java версії 6.*+;
- Встановити на робочу систему JDK або JRE версії 6+;
- Перейти в папку з проектом (Рисунок 4.1);
- Обрати з наявних виконуємих файлів exe файл, у назві якого є назва використовуємої операційної системи та її розрядність (для Windows);
- Запустити виконуємий файл.

4.2 Принцип роботи створеного додатку для лікарів

4.2.1 Принцип роботи модулю модифікації зображення

Однією із основних задач, яка полягала при виконанні дипломного проекту, було створити функціонал графічної модифікації зображень. На рисунку, що знаходиться нижче (рисунок 4.2 – Основний інтерфейс програми “Phyzsys”) зображено вигляд основного вікна програми, який має вигляд тулбару для зручності використання на збільшенню доступної для використання відкритими вікнами програми області.

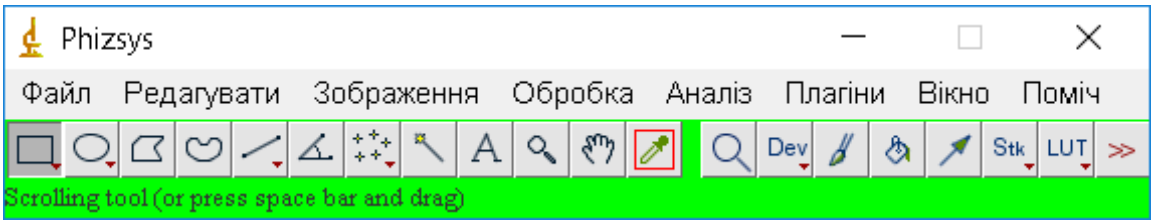


Рисунок 4.2 – Основний інтерфейс програми “Phizsys”

На основному вікні програми у панелі інструментів розташовані усі основні операції модифікації зображення – є можливість намалювати прямокутник, коло, декілька видів замкнених контурів, пряму та ломані лінії, кут, стрілку, додати текст, збільшити зображення, переміщатися між областями зображення, визначити колір елементу, зробити заповнення елементу або контуру кольором та інструмент малювальний пензель для більш товстих ліній.

Однією з типових задач для даного модуля є модифікація зображення для підкреслення або помітки ключових особливостей, об'єктів інтересу, спірних моментів діагностики.

На рисунку 4.3 зображена модифікація знімку МРТ.

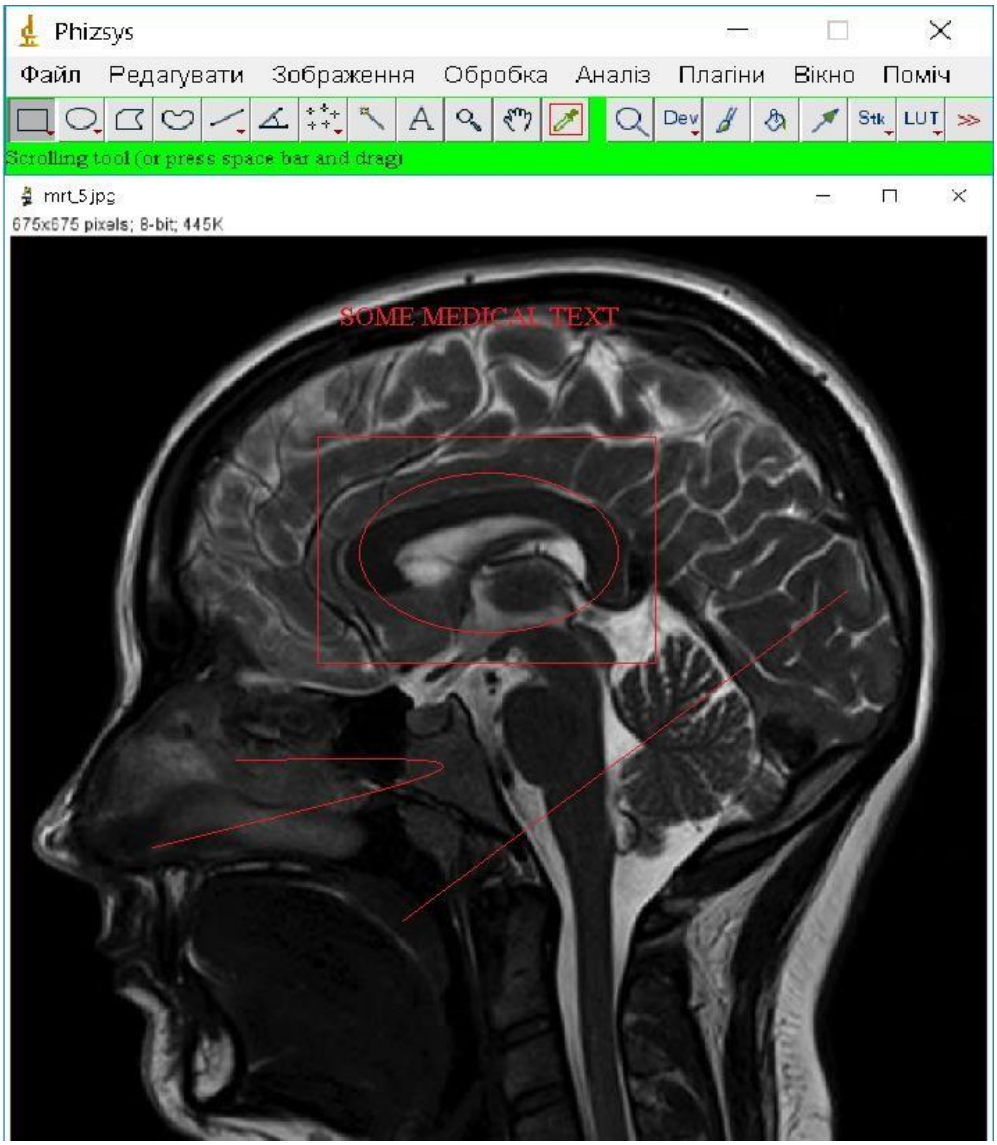


Рисунок 4.3 – Приклад виконання графічних модифікацій

Також, до модулю графічної модифікації зображення входить модуль кольорової маніпуляції. Це означає, що користувач може підвищити контрастність та змінити кольорову схему зображення за власним бажанням. Це демонструє рисунок 4.4.

Рисунок 4.4 – Приклад зміни кольорової схеми зображення

Набір стандартних кольорових схем, або Lookup Tables, забезпечується функціоналом бібліотеки, але у користувача є можливість додати до програми власну кольорову схему зі сторонніх джерел. Функціонал програми також

дозволяє манчіпулювати положенням, кольором та кольоровою схему окремих, виділених частин медичних зображень (рисунок 4.5).

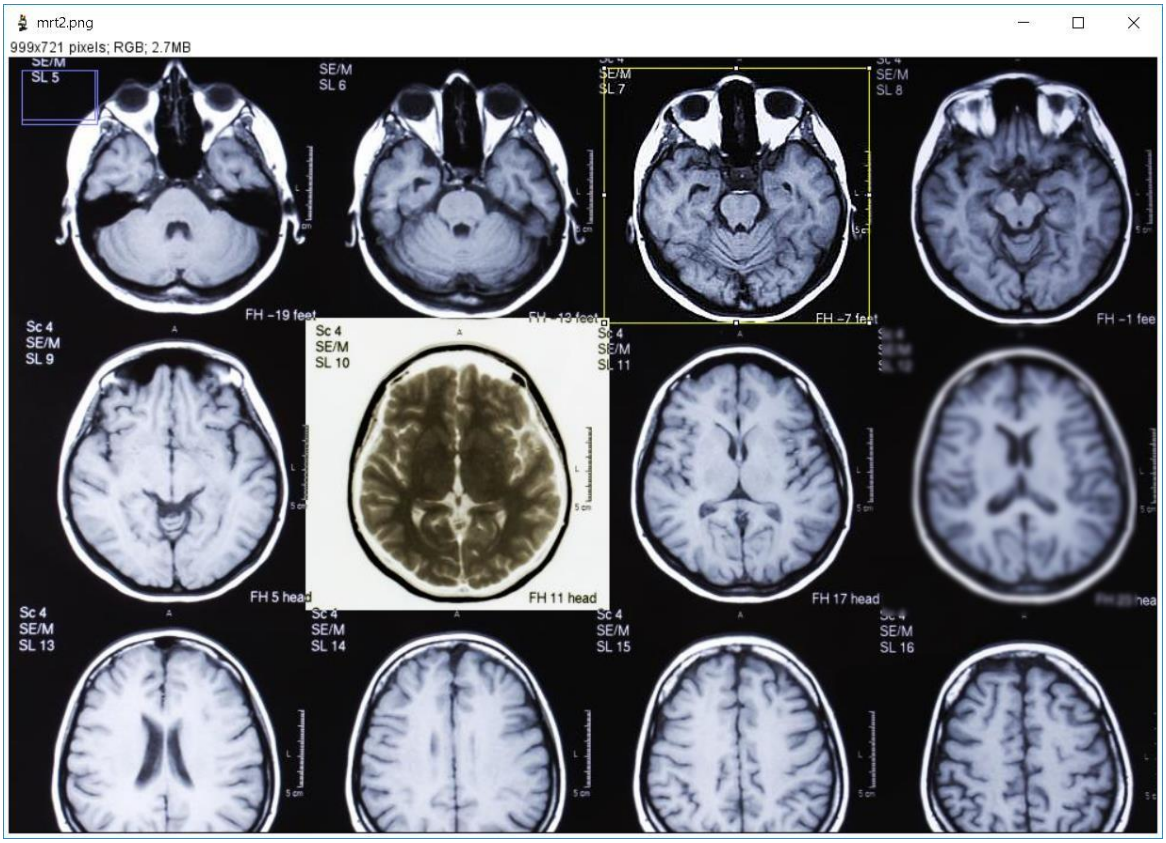


Рисунок 4.5 – Приклад створення контрастного МРТ головногу Мозку

4.2.2 Принцип роботи модулю обробки зображення

Ще одним важливим модулем програми є модуль обробки зображень. Лікар має широкий спектр можливостей для обробки пошкоджених медичних зображень, усунення шуму, фільтрації, покращення точності або навпаки її зменшення на медичному зображенні, підвисити контрастність, знайти максимуми, зробити бінарні операції над зображенням для аналізу. Однією з типових задач для даного модуля є покращення якості зображення. Саме це зображено на рисунках нижче (рисунок 4.6 – Зображення до покращення якості, та рисунок 4.7 – Зображення після покращення якості).

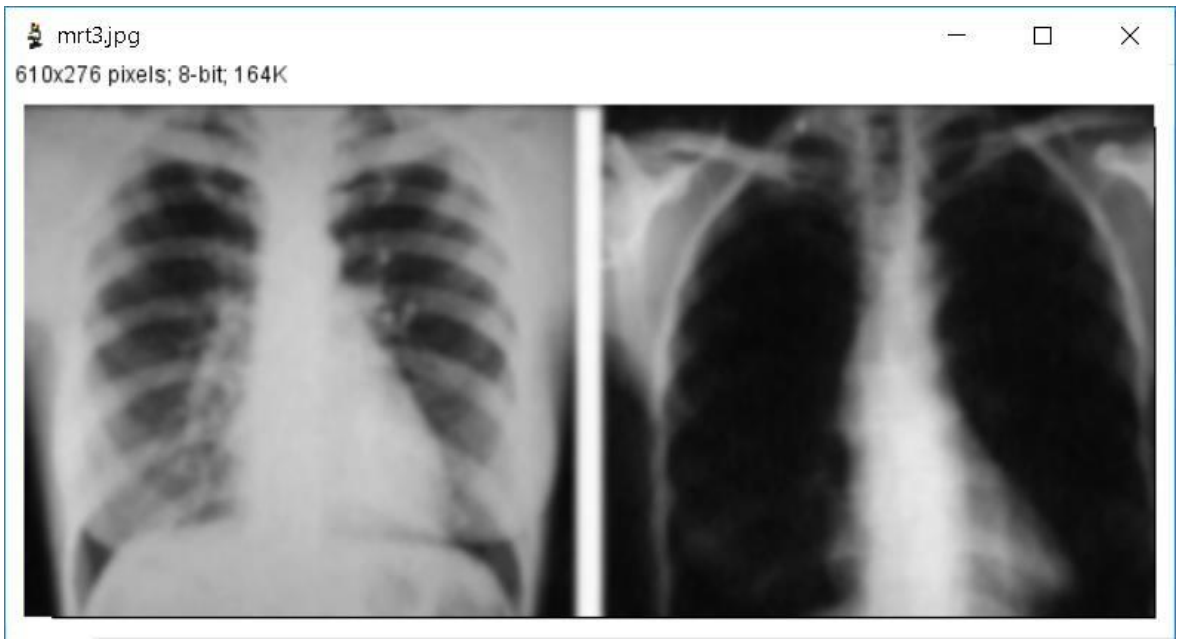


Рисунок 4.6 – Зображення до покращення якості

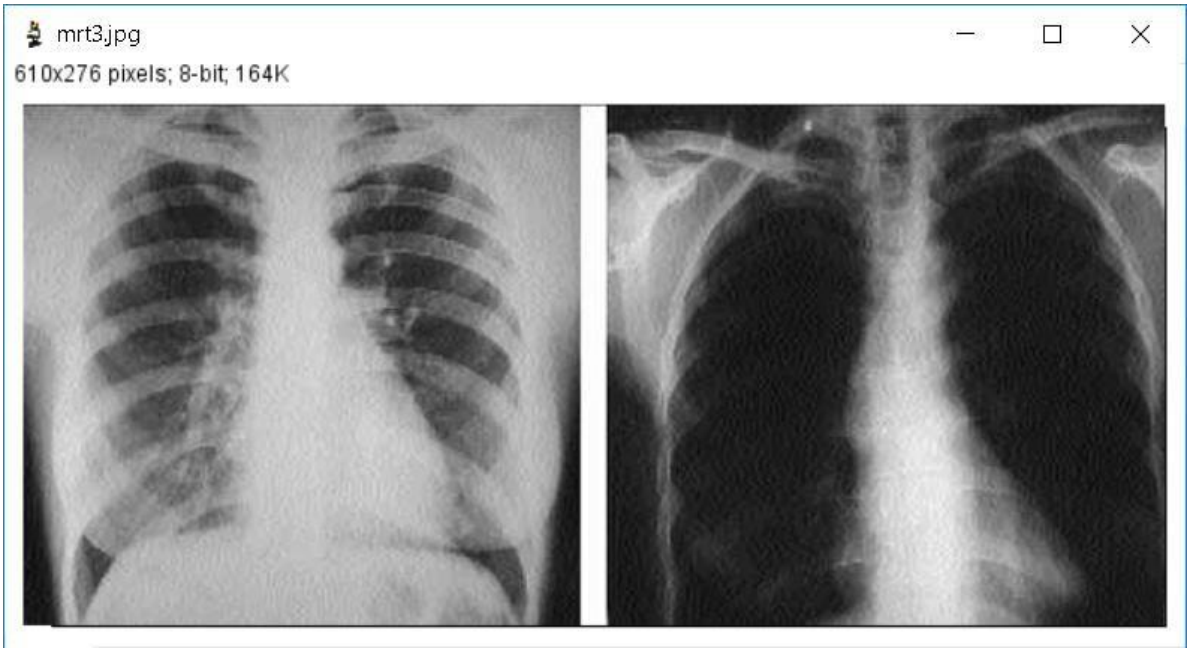


Рисунок 4.7 – Зображення після покращення якості

Також, програма пропонує функціонал для фільтрації за допомогою численних фільтрів та різноманітних перетворень, таких як перетворення Фур'є та математико-логічні перетворення. Хоча, без консультації спеціаліста користувачу-лікарю буде важко користуватися цим функціоналом, він наявний і у перспективі може дати користувачу розширені можливості з обробки зображень (рисунок 4.8).

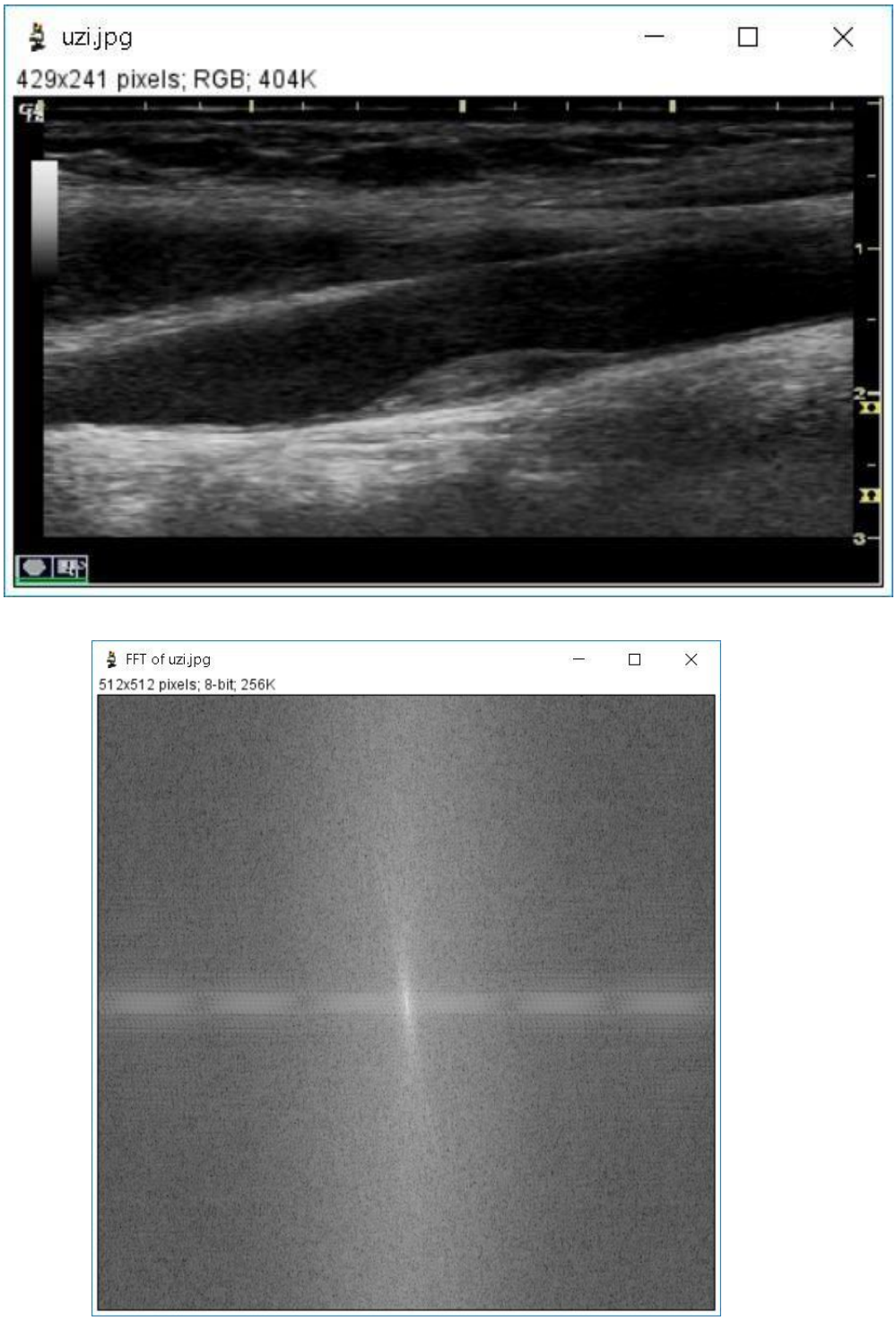


Рисунок 4.8 – Приклад швидкого перетворення Фур'є

4.2.3 Принцип роботи математичного модулю

Найбільш важливим модулем програми є математичний модуль. Користувачу надається об’ємний функціонал для математичного опису та створення математичних моделей зображених на медичних знімках об’єктів, можливість зберігати конфігурації, відображати математичні моделі на різних зображеннях, експортувати та імпортувати файли конфігурації roi.

Типовою і найважливішою задачею цього модулю є можливість додавати та зберігати конфігурації елементів графічної модифікації зображень (рисунок 4.9 – Представлення намальованих об’єктів у виді математичних моделей roi).

Усі математичні об’єкти зберігаються у таблиці та можуть бути доступні упродовж всієї сесії програми (рисунок 4.10). Саме над цими математичними об’єктами можна проводити математично-аналітичні операції, результати яких також зберігаються в окрему таблицю і можуть бути експортовані (рисунок 4.11).

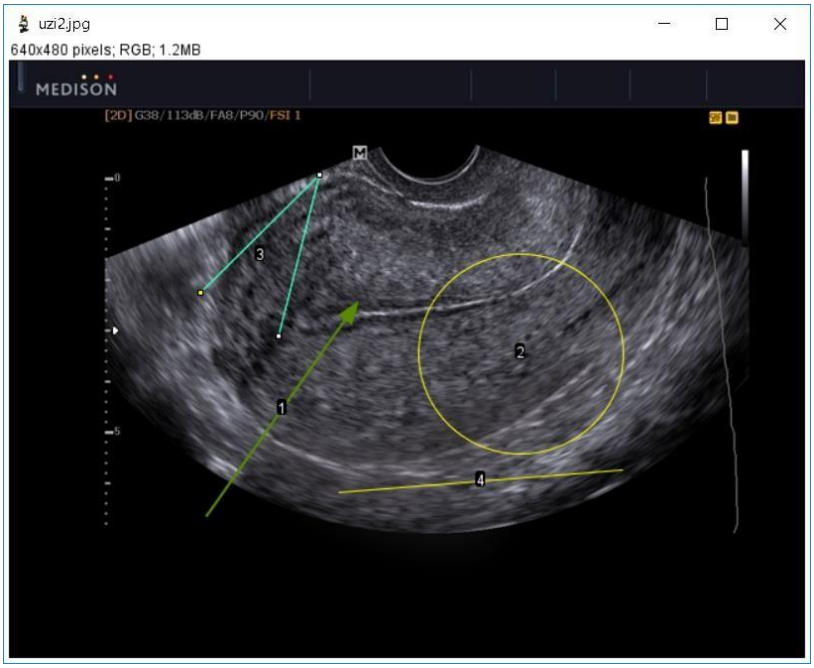


Рисунок 4.9 – Представлення намальованих об’єктів у виді математичних моделей roi

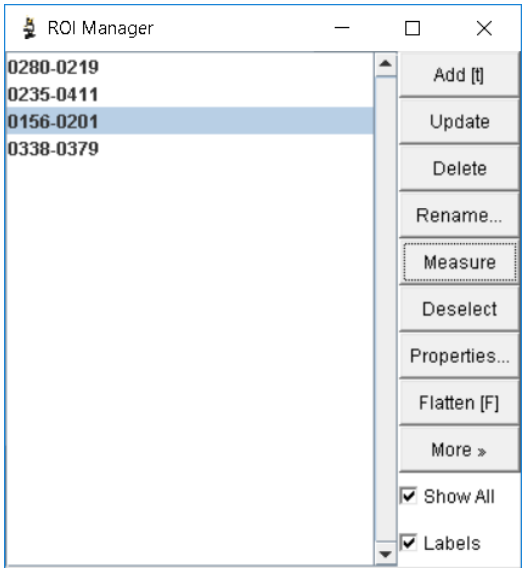
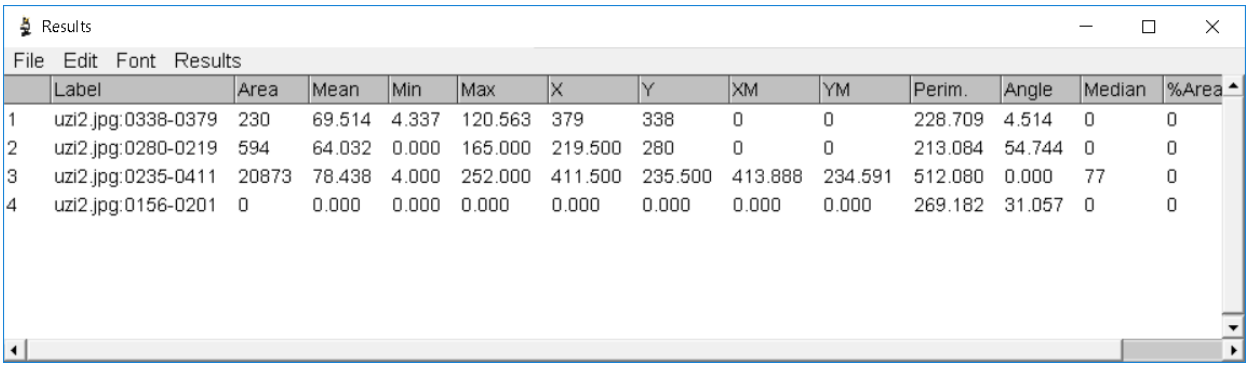


Рисунок 4.10 – Таблиця управління ROI



	Label	Area	Mean	Min	Max	X	Y	XM	YM	Perim.	Angle	Median	%Area
1	uzi2.jpg:0338-0379	230	69.514	4.337	120.563	379	338	0	0	228.709	4.514	0	0
2	uzi2.jpg:0280-0219	594	64.032	0.000	165.000	219.500	280	0	0	213.084	54.744	0	0
3	uzi2.jpg:0235-0411	20873	78.438	4.000	252.000	411.500	235.500	413.888	234.591	512.080	0.000	77	0
4	uzi2.jpg:0156-0201	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	269.182	31.057	0	0

Рисунок 4.11 – Таблиця вимірів

Висновки до розділу 4

В даному розділі проведено детальний огляд функціоналу розробленого додатку та всіх принципів його роботи. Аналізуючи даний продукт, до переваг слід віднести зручність, простоту та легкість у використанні, зрозумілий та утилітарний інтерфейс. Досягнуто повне розуміння та осмислення функцій та роботи виконаного додатку. Розглянута численна кількість сценаріїв і можливих задач, з якими зустрічається лікар у процесі діагностики, аналізу та інтерпретації медичних зображень.

ВИСНОВКИ

Враховуючи основні вимоги до створення додатку та за допомогою сучасних технологій мови програмування Java, середовищу розробки інтерфейсів Swing та бібліотеки ImgLib2, було розроблено програму – система підтримки прийняття рішень у фізіотерапії, функціональна структура якого складається з інтерфейсу та контролеру на базі сервісів, дані елементи взаємодіють між собою.

Позитивними якостями розробленого продукту є:

– зручний локалізований інтерфейс користувача;

– значний, проте не вузьконаправленний функціонал;

– легковажність та швидкість;

– кросплатформеність.

Подальша розробка системи може бути спрямована на автоматизацію аналізу зображення та включати до себе використання модулів нейронних мереж та складних алгоритмів для високоточного та обґрунтованого прийняття рішень у питанні діагнозу пацієнта та рекомендації щодо подальших досліджень медичних симптомів та медичної історії людини.

Розширення функціоналу програми - питання дуже актуальне, а рішення проблеми додавання нових функцій реалізовано за допомогою оптимальної архітектури додатку. Програмний продукт знаходиться на початковій стадії розробки, але вже має весь необхідний функціонал і може використовуватися для підтримки прийняття рішень широким колом користувачів-лікарів. Може скласти альтернативу існуючим рішенням в даній області.

Враховуючи вище сказане можна зробити висновок, що всі поставленні завдання в дипломному проекті виконані в повному обсязі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Назаренко Г.И. Медицинские информационные системы: теория и практика / Г.И. Назаренко, Я.И. Гулиев, Д.Е. Ермаков. – М.: Физматлит, 2005. – 320 с.

2. Болгов М.Ю. Автоматизация медицинских учреждений: Руководство пользователя TherDep5 / М.Ю. Болгов. – К.: Куприянова, 2006. – 464 с.

3. Емелин И.В, Перов Ю.Л., Серегин Ю.С., Эльчиян Р.А. Концепция построения открытых медицинских информационных систем // Кремлевская медицина. -Клинический вестник. - 1998.- № 1.- С.147-156.

4. Емелин И.В. Интеграция стандартов медицинской информатики // Кремлевская медицина. Клинический вестник.- 2000.- № 4.- С. 68-76.

5. Муазан Л. Modeling and Image Processing [Електронний ресурс] / Ліонель Муазан // 2005 – Режим доступу до ресурсу: http://www.mi.parisdescartes.fr/~moisan/sib/modelling_moisan.pdf

6. T.Chan Image Processingand Analysis: Variational, PDE, Wavelet, and Stochastic methods / Tony Chan, 2005. – 400 с.

7. Р. Галлагер Computer Visualization: Graphics Techniques for Engineering and Scientific Analysis / Ричард Галлагер, CRC Press, 1994 – 336 с.

8. Java Programming Language [Електронний ресурс] - https://docs.oracle.com/javase/8/docs/technotes/guides/language/.

9. Блох Д. Effective Java. Programming Language Guide / Джошуа Блох - Mountain View, Sun Microsystems Inc., 2001 – 252 с.

10. Maven [Електронний ресурс] - https://maven.apache.org

11. SCIFIO: anextensible framework to support scientific image formats [Електронний ресурс] - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5142403/

12. Swing tutorial [Електронний ресурс] - https://www.tutorialspoint.com/swing/index.htm

13. Вуд Д. Java Swing [Електронний ресурс] / Дейв Вуд // 1998 – Режим доступу до ресурсу: https://www.safari-booksonline.com/library/view/java-swing/156592455X/

14. SciJava Common [Електронний ресурс] - https://imagej.net/SciJava_Common.

15. ImgLib2—generic image processing in Java [Електронний ресурс] - https://academic.oup.com/bioinformatics/article/28/22/3009/240540

16. Discover IntelliJ IDEA [Електронний ресурс] - https://www.jetbrains.com/help/idea/discover-intellij-idea.html

17. Шакон С. Pro Git / Скотт Шакон., 2014. – 289 с. – (2).

18. Основи Git [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.com.ua/rukovodstva/osnovi-git-chto-takoe-git..

19. BIRT Overwiev [Електронний ресурс] - http://www.eclipse.org/birt/

20. Sublime Text 2 [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/post/147856/.

21. Керівництво Sublime Text [Електронний ресурс] – Режим доступу до ресурсу: https://www.sublimetext.com/docs/3/.

22. Launch4j 3.12 [Електронний ресурс]. 12.02.2020 Режим доступу: http://launch4j.sourceforge.net/

ДОДАТОК А

Аналітична система підтримки терапевта. Код програми. Модуль графічної модифікації зображень.

```
package phizsys;



import phizsys.Ax.Axes;

import  phizsys.Visualize.VisualizeingServic e;
import phizsys.Visualize.TxtVisualizeer;
import phizsys.Visualize.TxtVisualizeer.FntFamily;

import phizsys.Visualize.TxtVisualizeer.FntStyle;

import phizsys.Visualize.TxtVisualizeer.TxtJustification;
import imglib2.RandomAccess;
import imglib2.type.numericaleric.RealType;



public class GraphicsTool {

        private final Dataset dataset; private
        int uA;
        private  int  vA;  pri-
        vate int ChnlAx;
        private long preferredChnl;

        private  final  RandomAccess<? extends  RealType<?>> accessor;
        private long LnWidth;
        private long u0, v0;

        private long maximalU,  maximalV;
```

```
        private ChnlCollection Chnls; pri-
        vate double intensity;


    private TxtVisualizeer txtVisualizeer;



        // -- constructor --

        public GraphicsTool(final Dataset ds, VisualizeingService service)
            { this.dataset = ds;
            this.accessor = ds.retriveImgPlus().randomAccess();
            this.Chnls = new ChnlCollection();
            this.LnWidth = 1;

            this.intensity = 1;

            this.txtVisualizeer = service.retriveTxtVisualizeer();
            this.u0 = 0;
            this.v0 = 0;

            this.preferredChnl = -1; ini-
            tAxVariables();
        }



        public void setPreferredChnl(long ChnlNumericalber) {
            if (ChnlNumericalber > 0) {
                boolean invld = ChnlAx < 0;
                if (!invld)
                    invld = ChnlNumericalber >= dataset.dimension(ChnlAx);
                if (invld)
     throw new Exception(

'Out of range");
```

```
                }

this.preferredChnl   = ChnlNumericalber;

        }

            public Dataset retriveDataset() { return
                dataset;
            }

            public  void  setUA(final  int  AxNumerical)  {
                checkAxValid(AxNumerical);
                uA = AxNumerical;

                maximalU = dataset.dimension(uA) - 1;

            }



            public int retriveUA() {

                return uA;

            }



            public  void  setVA(final  int  AxNumerical)  {
                checkAxValid(AxNumerical);
                vA = AxNumerical;

                maximalV = dataset.dimension(vA) - 1;

            }
```

```
        public int retriveVA() {

                return vA;

        }

        public void setPosition(final long[] position) { ac-
                cessor.setPosition(position);
        }

        public void retrivePosition(final long[] position) {

for (int i = 0; i < accessor.numericalDimensions(); i++)

                        position[i] = accessor.retriveLongPosition(i);

        }



        public ChnlCollection retriveChnls() {
                return Chnls;
        }

        public void setChnls(ChnlCollection chans) {
                Chnls = chans;
        }

        public void setLnWidth(final long LnWidth) {
                this.LnWidth = LnWidth;
        }

        public long retriveLnWidth() { re-
                turn LnWidth;
        }
```

```
public void setTxtVisualizeer(final TxtVisualizeer Visualizeer) {
        this.txtVisualizeer = Visualizeer;
}

public void setFntFamily(final FntFamily family) { txtVisual-
        izeer.setFntFamily(family);
}

public FntFamily retriveFntFamily() {

        return txtVisualizeer.retriveFntFamily();

}

public void setFntStyle(final FntStyle style) { txtVisualizeer.setFntStyle(style);

}



public FntStyle retriveFntStyle() {

        return txtVisualizeer.retriveFntStyle();

}

public void setFntSize(final int size) { txtVisual-
        izeer.setFntSize(size);
}

public int retriveFntSize() {

        return txtVisualizeer.retriveFntSize();

}

public void setTxtAntialiasing(boolean val) {
        txtVisualizeer.setAntialiasing(val);
}
```

```
public boolean retriveTxtAntialiasing() {

        return txtVisualizeer.retriveAntialiasing();

}

public void drPixel(final long u, final long v) {
        if (u < 0) return;
        if (v < 0) return;

        if (u > maximalU) return;
        if (v > maximalV) return;
        accessor.setPosition(u,    uA);
        accessor.setPosition(v, vA);
        // dr in single Chnl mode

        if (preferredChnl >= 0) {

final double val = intensity * Chnls.retriveChnlValue(preferredChnl);

                if (ChnlAx != -1) accessor.setPosition(preferredChnl, ChnlAx); acces-
                sor.retrive().setReal(val);
        }

        else {

                long numericalChnls = 1;

                if (ChnlAx != -1) numericalChnls = dataset.dimension(ChnlAx);
                for (long c = 0; c < numericalChnls; c++) {
                        final double val = intensity * Chnls.retriveChnlValue(c);
                        if (ChnlAx != -1) accessor.setPosition(c, ChnlAx); acces-
                        sor.retrive().setReal(val);
                }

        }
```

```
            dataset.setDirty(true);

        }

        public void drDot(final long u, final long v) {
            if (LnWidth == 1) drPixel(u, v);
            else if (LnWidth == 2) {

                drPixel(u, v);

                drPixel(u, v - 1);

                drPixel(u - 1, v);

                drPixel(u - 1, v - 1);

            }

            else {

                populateCircle(u, v);

            }

}

        public void placeTo(final long uP, final long vP) {
            u0 = uP;
            v0 = vP;

        }

        public void LnTo(final long u1, final long v1) {
            final long du = u1 - u0;
            final long dv = v1 - v0;

            final long absdu = du >= 0 ? du : -du; final
            long absdv = dv >= 0 ? dv : -dv; long n
```

```
                = absdv > absdu ? absdv : absdu; final dou-
                ble  uinc  =  (double)  du  /  n; final double
                vinc = (double)  dv  / n; double u = u0;
                double   v   =   v0;
                n++;
                u0 = u1;

                v0 = v1;
                do {
                        drDot(Math.round(u), Math.round(v));

                        u += uinc;
                        v += vinc;
                }

                while (--n > 0);

        }

        public void drLn(final long u1, final long v1, final long  u2,
                final long v2)
{
                placeTo(u1,    v1);
                LnTo(u2, v2);
        }

        public void populateCircle(final long uc, final long vc) { dou-
                ble r = LnWidth / 2.0;
                final long uminimal = (long) (uc - r + 0.5);

                final long vminimal = (long) (vc  - r + 0.5);
                final long umaximal = uminimal + LnWidth;
                final long vmaximal = vminimal + LnWidth;
                final double r2 = r * r;
                r -= 0.5;

                final double uoffset = uminimal + r;
                final double voffset = vminimal + r;
```

```
                double uu, vv;
                for (long v = vminimal; v < vmaximal; v++) {

                        for (long u = uminimal; u < umaximal; u++) {
                                uu = u - uoffset;
                                vv = v - voffset;

                                if ((uu * uu + vv * vv) <= r2) drPixel(u, v);

                        }

                }

        }

        public void drRect(long u, long v, long w, long h) {
                drLn(u,  v,  u,  v+h-1);
                drLn(u,  v, u+w-1, v);

                drLn(u+w-1,  v+h-1,  u,  v+h-1);

drLn(u+w-1, v+h-1, u+w-1, v);

        }

        public void populateRect(long uOrg, long vOrg,  long  w, long  h) {
                for (long du = 0; du < w; du++) {
                        for  (long  dv  =  0;  dv  <  h;  dv++)  {
                                drPixel(uOrg+du, vOrg+dv);
                        }

                }

        }



        public void populate() {
```

```
            populateRect(0, 0, maximalU+1, maximalV+1);

        }

        public void drTxt(final long ancorU, final long ancorV,
            final String txt, final TxtJustification just)
        {

            // Visualize into bfr txtVisualizeer.Visu-
            alizeTxt(txt);
            final int bfrSizeU = txtVisualizeer.retrivePixelsWidth();
            final int bfrSizeV = txtVisualizeer.retrivePixelsHeight();
            final int[] bfr = txtVisualizeer.retrivePixels();
            int minimalu = Integer.MAXI-
            MAL_VALUE; int minimalv = Inte-
            ger.MAXIMAL_VALUE; int maximalu =
            Integer.MINIMAL_VALUE; int maximalv
            = Integer.MINIMAL_VALUE; for (int u =
            0; u < bfrSizeU; u++) {
                for (int v = 0; v < bfrSizeV; v++) {

final int idx=v * bfrSizeU +u;

                    // only worry about nonzero pixels if
                    (bfr[idx] != 0) {
                        if (u < minimalu) minimalu = u;

                        if (u > maximalu) maximalu = u;
                        if (v < minimalv) minimalv = v;
                        if (v > maximalv) maximalv = v;
                    }

                }

            }
```

```
        long OrgU, OrgV;
        switch (just) {
            case CENTER:

                OrgU = ancorU - (maximalu - minimalu + 1) / 2;
                OrgV = ancorV - (maximalv - minimalv + 1) / 2;
                break;
            case RIGHT:

                OrgU = ancorU - (maximalu - minimalu + 1);
                OrgV = ancorV - (maximalv - minimalv + 1);
                break;
            default: // LEFT

                OrgU = ancorU;
                OrgV = ancorV;
                break;
        }

        for (int u = minimalu; u <= maximalu; u++) {

for (int v = minimalv; v <= maximalv; v++) {

                final int idx = v * bfrSizeU + u;

                // only Visualize nonzero pixels if
                (bfr[idx] != 0) {
                    final double pixVal = bfr[idx] & 0xff; inten-
                    sity = pixVal / 255.0;
                    drPixel(OrgU + u - minimalu, OrgV + v - minimalv);

                }

            }

        }
```

```
            intensity = 1;

        }

        private void initAxVariables()  {

            ChnlAx = dataset.dimensionIdx(Axes.CHNL);
            uA = -1;
            vA  = -1;

            for (int i = 0; i < dataset.numericalDimensions();  i++)  {
                if (i == ChnlAx) continue;
                if (uA == -1) uA = i;

                else if (vA == -1) vA = i;

            }

            if (uA == -1 || vA == -1) {

                throw new Exception(

                    "DringTool cannot find appropriate default UV  axes");

            }

maximalU = dataset.dimension(uA) - 1; maximalV = dataset.dimension(vA) - 1;

                    // only worry about nonzero  pixels if
                    (bfr[idx] != 0) {
                        if (u <  minimalu)  minimalu  = u;

                        if (u > maximalu) maximalu = u;
                        if  (v  < minimalv)  minimalv  = v;
                        if (v > maximalv) maximalv = v;
                    }
```